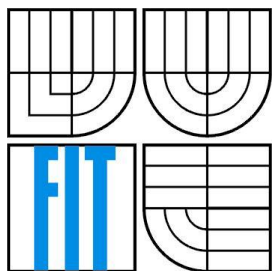


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

NOVÝ ÚSVIT POJMENOVÁVÁNÍ, ADRESOVÁNÍ A SMĚROVÁNÍ NA INTERNETU

A NEW DAWN OF NAMING, ADDRESSING AND ROUTING ON THE INTERNET

DIZERTAČNÍ PRÁCE
DISSERTATION THESIS

AUTOR PRÁCE
AUTHOR

Ing. Vladimír VESELÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. Miroslav ŠVÉDA, CSc.

BRNO 2013 – 2015

Abstrakt

Internet roku 2015 se potýká s problémy, které jsou důsledky špatného designu pojmenování a adresování v TCP/IP a jež mají přeneseny vliv i na škálovatelnost směrování. Problémy jako růst páteřních směrovacích tabulek, neefektivní multihoming sítě či mobilita zařízení a mnohé další zadávají k otázce, jestli není třeba architekturu Internetu pozměnit. V teoretické části je kvantifikován dopad problémů, možná řešení a zejména je formálně definována teorie kompilující poznatky významných publikací zabývajících se problematikou pojmenování, adresování a směrování v počítačových sítích. Tato práce se zabývá dvěma konkrétními technologiemi, jež mají ambicí Internet měnit - Locator/Id Separation Protocol a Recursive InterNetwork Architecture. Výstupem práce jsou vylepšení funkcionality obou výše zmíněných technologií. Za účelem praktického ověření dopadů našeho výzkumu jsou vyvinuty a popsány nové simulační modely pro OMNeT++, které jsou věrné úrovni detailu popisu ze specifikací.

Klíčová slova

Internetová architektura, pojmenování a adresování, směrování, oddělení lokátorů a identifikátorů, LISP, rekurzivní mezisíťová architektura, RINA, OMNeT++

Abstract

Internet of the year 2015 struggles with problems that are just implications of flawed naming and addressing the concept of TCP/IP, which have an impact on overall routing scalability. Problems such as default-free zone routing table growth, cumbersome multihoming or mobility motivate question whether the Internet deserves major architecture redesign. In the theoretical part, the impact of problems above is evaluated, solutions are discussed and unifying theory compiled and described using formal methods taking into account revered papers about naming, addressing and routing. This work provides in-depth investigation of two technologies - Locator/Id Separation Protocol and Recursive InterNetwork Architecture. Research contribution is an operational improvement of above-mentioned technologies. New OMNeT++, full-fledged simulation modules compliant with behavior in the specification are used to as verification tool.

Keywords

Internet architecture, naming and addressing, routing, locator/id split, LISP, Recursive InterNetwork Architecture, RINA, OMNeT++

Poděkování / Acknowledgement

It took seven years of my life to finish this thesis during which I probably became one of the oldest still studying Ph.D. students in the Czech Republic. There is a legion of people without whom I would never succeed. I would like to thank them all! If you are a reviewer, defense committee member or knowledge seeker, then you can skip this part, otherwise be understanding and enjoy. For a moment, I will switch back to my native language because largest group of astute readers of this section would not appreciate English acknowledgment.

Posledních sedm let bylo nesmírně krásným obdobím mého života, ve kterém jsem dostal šanci poznat, jaké to je být akademikem, jaké to je být vědcem. Podařilo se mi za tu dobu nahlédnout a prozkoumat něco málo v oboru počítačových sítí (oboru, který majoritně ovlivňuje Internet, jeden z nejúžasnějších lidských vynálezů).

Má první a největší slova díky patří mé rodině. Rád bych poděkoval svým rodičům, mamince Jeleně a tatínkovi Vladimírovi, za svůj život, protože bez jejich lásky a podpory bych se nikdy nestal tím, kým jsem, a nedostal tam, kde jsem. Chtěl bych poděkovat i svému bráchrvi Ondrovi za báječné sourozenectví; bez jeho vzorného příkladu a hravosti bych nikdy nenabyl všech klíčových vlastností, které jsou nyní esenciálními v mém životě. Poděkování patří i mým prarodičům: babičce Evě a dědečkovi Macíkovi za pečlivost, cílevědomost a za to, že mi pomáhají ukazovat, co je v životě opravdu důležité; babičce Květě a dědečkovi Kulíškovi za vrozenou zvědavost a pochopení síly lidské dobroty. Okruh rodinný uzavírá a bezezbytku doplňuje Marie – vyšší bytost, které jsem zavalil své srdce. Marie, která je mi stabilním vorem na rozbouřeném moři života a která je mou vůbec nejvíc nelepčejší čupr kamarádkou. S radostnou novinou o dopsání této práce se podělím i s mou širší rodinou – tetě Evě + strýci Laděovi, tetě Daně + strýci Borkovi, sestřenicím Danuše + Jituše, bratrancům Honzovi + Adamovi, švagrové Veronice, synovcům Radimkovi + Honzíkovi, adoptivním doktorským tetičkám Vlastě, Blance, Jarce a Lidce a adoptivnímu strejdovi Mirkovi – nepřestávám být vděčný, že všechny ty krásné duše které byly, jsou a budou navždy součástí právě mého života.

Nyní budou následovat všichni ti, kteří se fakticky zasloužili za můj profesní rozvoj. Bez možnosti konfrontovat s nimi svou (ne)znalost, by konkrétně tato disertační práce skutečně nikdy nevznikla. V první řadě nesmírně díky Mirku Švédovi, tomu nejhodnějšímu Panu Profesorovi, který je schopen vás chránit a jak lev se za vás bít i ve chvílích, kdy si to možná ani nezasloužíte. Dále bych chtěl vyjádřit nesmírný vděk Ondrovi Ryšavému a Petrovi Matouškovi, školitelům-specialistům, kteří mě svými radami a vedením dokázali vždy postrčit správným směrem v mé akademicko-vědecké kariéře. Zejména pak obzvláštní díky dr./doc. Ryšavému za pomoc a revizi s mnohými pasážemi. Nebýt Matěje Grégra (± Karolíny Hlobilové) a Svatopluka Šperky, tak bych si nikdy neudržel po ty roky tak dobrou, nutno však podotknout že ne pracovní morálku. Díky sedánkům s nimi nad pivem a kávou jsem byl schopen poznat míru své neznalosti, protože pokud nějaký IT problém nejste schopni vysvětlit ani svým přátelům z oboru, tak tomu prostě nerozumíte. Liboru Polčákovi za rady a záviděníhodné pracovní

prostředí v kanceláři. RINASim by zůstal jen výplodem fantazie, kdyby nebylo mých spolehlivých a pracovitých kolegů Marcela Marka, Tomáše Hykla a Kamila Jeřábka. ANSAINET jako projekt žije a pokračuje jen díky nasazení šikovných diplomantů, jako byli Vladimír Kojecký, Zdeněk Kraus, Marek Černý, Veronika Rybová, Matej Hrnčířik, Jakub Smejkal, Jakub Mrázek, Tomáš Procházka, Jiří Trhlík, Adam Malik, Petr Vítek, Jan Bloudíček a Vít Rek. Nikdy bych se sítěmi nezabýval tak horlivě a s takovým zápalem, kdybych hned na začátku nepoznal opravdové mistry v tomto oboru, Michala Rapca CCIE R&S #18608, Igora Foulda CCIE Sec #20135 a Petera Palúcha CCIE R&S #23527, kterým vděčím za svůj osobní rozvoj. Na závěr bych pár slov díky věnoval i paní Sylvě Sadovské za ochotu a báječné nasazení v poslední fázi realizace dizertace.

K nefyzickým adorantům pak patří čtyři entity: 1) výzkumná skupina NES@FIT, která je schopna lákat skvělé vědecké naděje, s nimiž je radost spolupracovat; 2) Ústav informačních systémů (vedené doc. Dušanem Kolářem a organizované péčí paní Michaely Bílkové) za vstřícnost a poskytování záviděníhodného zázemí; 3) alma mater Fakulta informačních technologií Vysokého učení technického v Brně, ze všech institucí vysokých škol, na kterých jsem měl tu možnost studovat, je ona tou nejlepší a nejpřednější; a 4) Městský úřad v Šumperku za nehorázně dlouhé fronty, kde čekání v nich dalo vzniknout těm nejkritičtějším pasážím této práce.

Nyní bude následovat skupina živočichů hodně mi blízkých leč bez přímého dopadu na můj profesní život. V šiku na první řadě stojí mí tři nejbližší přátelé Karel Záruba, Jan Beránek a Rostislav Pumprla; i přes roky, co se známe, mě neustále překvapuje, že jsme se zatím nepovraždili. S těmito dvěma a ještě s Janem Bělinem (+Janou Hlávkovou), Martinem Zárubou a Martinem Ptaškem (+Marií Kratochvílovou) mám tu nesmírnou čest čas od času bavit publikum na prknech (která znamenají ochotnický svět) porcí kvalitní zábavy odpovídající jménu kumpanie Děs/Běs. A díky Martě Fišerové a Marcínu Cwiklińskému mám z bezprostřední blízkosti možnost pozorovat chování a zvyklosti druhu *Homo sapiens artis* v jeho přirozeném prostředí prostor Galerie Klubovna. Za přátelství překonávající vzdálenosti a léta pak:

- Janu (který chápe, jak je důležité mít nízké THAC0) a Báře (za nevídanou znalost v problematice parfémů) Sporkovým, Karlu (mému ktkvi) a Kláře Hoškovým, Evě Strnadové (za to, že je pořád stejná) + Ondru Vávrovi, Martinu (že se nás od Majálesu nepustil) a Radce Kalbáčovým, Vojtu Beilovi (opravdovému gentlemanovi) + Ivě Hlavěnkové, Ludku (za poznatky o letadlech) a Jarce (za poznatky o mateřství) Krmelovým, Kláře Krkonoškové (za cestovní dobrodružství), Jorhu Akritidisovi (za to, že se dá vždycky vstát, když spadnete), Ireně Valertové (že je hlasitější než já), Evě Suchánkové, Magdě Suché, Markétě Suchánkové (za nopakovatelné party v domě jejich rodičů), Martině Klusákové;
- jesenickým Svatopluku Sejkorovi a Adamu Perutkovi a Patriku Pavlíčkovi (za stovky partií desítek her, co jsme spolu hráli);

- brněnským Agnieszce Landowské (za mateřství Alojze Bobka), Karlu a Ivaně Kotrbovým (za privat), Jaromíru a Evě Výtvarovým (za pískoviště), Stanislavu Židkovi (za první projekt do IOS a pomoc při laboratorních cvičeních z fyziky), Lence Jalůvkové a Jakubu Křoustkovi, Petře Hříbové (za překonání strachu k hadům), Tereze Kurovské (za pochopení síly slušnosti), Kateřině Novotné (za pohledy), Edwardu Robe’mu (for American Madness and American Hunger), Janě Šafářové (za odhodlání stát se, kým chcete), Elišce Kňobortové (za dopisy);
- měl jsem to štěstí a sílu zdolat již dvakrát (a pevně doufám, že ne naposledy) Svatojakubskou pouť do Santiaga de Compostely, kde na každé z cest se mi do srdce navždy otiskly bytosti jako Martina “Levandule” Havlíková (za dýňová semínka a pochopení, že Camino je život), SangGyu “Sol” Choi (za poznání zázraku splněného snu), Daniel Perloff (one day I will reach the beginning of Appalachian Trail), Grigory Petrenko, Kyril Pevnev a Denis (Приветствую моих сибирских друзьями. Это мой первый и наименее важная книга упомянуть тебе, Григорий! Я должен тебе пиво для долгой дружбы и приключений еще впереди).

Také bych rád poděkoval několika výjimečným zástupcům *Canis lupus f. familiaris* – mému psovi Atosovi, bearded-kólii jejíž smrt byla počáteční událostí mého PhD studia; Falkovi I., II. a III. za náklonnost k smečce Veselých; feně Maye, rhodesian ridgeback psu války +5; psovi Šotkovi, psychicky narušené pouliční směsce; feně Daisy, bez níž se neobešla žádná správná party – bytostem schopným rozdávat lásku jen za žrádlo a pohlazení.

Ve velké většině prací, kterou jsem měl tu radost vést, se v poděkování na doporučení vedoucího nachází (ne)povinně i recept na dobré jídlo. Bylo by ode mě tedy nanejvýš pokrytecké do této pozvolna vznikající kuchařky nepřispět. Dlouho jsem zvažoval, jaká kulinářská šmakuláda by to měla být, než jsem se rozhodl pro pokrm nejen chutný, ale i zdravý prospěšný. Zde tedy je algoritmus k přípravě chleba s česnekem: ① vezměte krajíc chleba; ② namažte ho vrstvou sádla (rodina Veselých) nebo másla (rodina Přikrylových); ③ posypejte ho dvěma stroužky česneku rozkrájenými na tenká „kolečka“; ④ osolte (gurmáni zvolí sůl bylinkovou); a nakonec ⑤ konzumujte bez zaváhání a bez uroněných slz. Tato potrava je vhodná jako rychlá svačinka před důležitou schůzkou, či jako lék proti nastuzení, nebo jako zaručená obrana při boji s upíry.

Moreover, now back to English to praise the last but not least group of people. My biggest and sincerest appreciation goes to John Day, who opened my eyes, showed me the history of computer networking and introduced me to other RINA fans. I sincerely treasure all the Socrates-like discussions with John that showed me the depth of my (computer networking) ignorance. I owe big acknowledgment to all PRISTINE consortium members for their trust and for letting Faculty of Information Technology – Brno University of Technology (FIT-BUT) to participate in something meaningful. Namely, I would like to thank Eduard Grasa and Jordi Perelló for being such patient and extraordinary reviewers.

Tato stránka je dedikováno autogramům a dalším podpisům...

© Vladimír VESELÝ, 2015

Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Contents

1	Introduction.....	1
2	Networking Fundamentals	3
2.1	Basic Terminology	4
2.2	Present Problems of Internet.....	10
2.2.1	Routing Scalability	11
2.2.2	Decoupling Identification and Location.....	14
2.2.3	Multihoming.....	15
2.2.4	Mobility.....	17
2.2.5	Traffic Engineering.....	18
2.2.6	Renumbering	18
2.3	Influencing Factors	20
2.3.1	Burden on Routing Table Size.....	20
2.3.2	Burden on Routing Table Processing.....	23
2.4	Chapter Summary	25
3	Naming and Addressing Concepts	26
3.1	Basic Terminology	27
3.2	Analogies.....	31
3.2.1	Naming and Addressing in Telephony.....	31
3.2.2	Naming and Addressing in Postal Service	32
3.2.3	Naming and Addressing in Operating Systems	32
3.3	Theory	34
3.4	Praxis.....	37
3.4.1	Internet Protocol Version 6	38
3.4.2	Domain Name System	40
3.4.3	Uniform Resource Identifier.....	41
3.5	Possible Solution.....	42
3.5.1	Ideal Solution Properties	42

3.5.2	Existing Proposals.....	43
3.5.3	Proposals Comparison.....	53
3.6	Chapter Summary	57
4	Locator/ID Separation Protocol	58
4.1	Overview	59
4.1.1	Tunneling	61
4.1.2	Mapping System	61
4.1.3	Coexistence between LISP and Non-LISP.....	66
4.2	LISP Demonstrations	68
4.2.1	Unicast Communication.....	68
4.2.2	Registration	69
4.2.3	Mapping Request	70
4.2.4	Mapping Reply	71
4.2.5	Proxy Communication	72
4.3	State-of-the-Art.....	74
4.3.1	Implementations	74
4.3.2	Deployment	75
4.3.3	Simulators	75
4.4	Contribution.....	76
4.4.1	Virtual Router Redundancy Protocol	76
4.4.2	Map-Cache Synchronization	78
4.4.3	Merged RLOC Probing	83
4.4.4	Design and Implementation.....	84
4.4.5	Results.....	89
4.5	Chapter Summary	97
5	Recursive Internet Architecture	98
5.1	Overview	99
5.1.1	Nature of Applications and Application Protocols	99
5.1.2	Core Terms	100

5.1.3	Connection-oriented vs. Connectionless	101
5.1.4	Delta-t Synchronization	102
5.1.5	Separation of Mechanism and Policy	102
5.1.6	Naming and Addressing	103
5.2	RINA Components	105
5.2.1	Nodes	105
5.2.2	Distributed Application Process Components	106
5.2.3	IPC Process Components	107
5.3	State-of-the-Art	120
5.3.1	Projects	120
5.3.2	Implementations	120
5.3.3	Simulators	120
5.4	Contribution	121
5.4.1	Installation	121
5.4.2	Design	121
5.4.3	RINASim Demonstration	129
5.5	Chapter Summary	141
6	Conclusion	142
6.1	Summary about LISP	143
6.2	Summary about RINA	144
6.3	Future Work	145
6.4	Final Thoughts	145
7	Bibliography	146
8	Addendum	158
8.1	Formats of LISP Control Messages	158
8.1.1	LISP Map-Request	158
8.1.2	LISP Map-Response	159
8.1.3	LISP Map-Register and LISP Map-Notify	160
8.2	ANSARouter Module	161

8.3	Additional Graphs	162
8.3.1	Map-Cache Sync Scenario with Single xTR1 Outage	162
8.3.2	Map-Cache Sync Scenario with Three xTR1 Outages.....	162
8.3.3	RLOC Probe Scenario with Eighty EIDs	163
8.4	John Day about RINA.....	164
8.5	RINASim Policies	165
8.6	RINASim Demonstration	167
8.6.1	omnetpp.ini.....	167
8.6.2	config.xml	168
8.6.3	EnrollmentStateTable Contents Progress	169
8.6.4	NFlowTable Contents Progress	170
9	Lists	171
9.1	Tables	171
9.2	Figures.....	172
9.3	Index.....	175

1 Introduction

☞ –“*Yesterday is gone. Tomorrow has not yet come. We have only today. Let us begin.*” Mother Teresa
☞ *What are goals and motivations of this thesis?*

Nowadays Internet *routing*, *naming* and *addressing* concepts (in the sense of common Internet-based computer networking audience) are facing a variety of challenges that were not so apparent in early days of the TCP/IP stack. Among those challenges, there are multihoming, mobility, traffic engineering, renumbering, **node** (a.k.a. **device**¹) localization and identification and routing scalability connected with the growth of the global routing tables.

IRTF’s RRG² was, and IETF’s IAB³ is for a long time in charge of observing trends in routing, collecting statistics and suggesting architectural recommendations influencing tendencies in future networking. In this thesis, we try to describe and evaluate the impact of these trends. Moreover, we gather relevant proposals and compare them to each other. Among documents and proposals discussed, there are also Locator/ID Separation Protocol (LISP) and Recursive Internet Architecture (RINA) that receive both positive and also negative reviews. We believe that both of them are addressing the same fundamental issues (which serve as the motivation behind our research), but they employ a very different approaches. While the first one is trying to repair the most apparent problems using existing architecture. The objective of the second one is to return the Internet to the original architecture model and generalize from there.

The LISP presents a new routing architecture based on the idea of splitting the device identity and the device location into two separate namespaces. Managing identity and location separately provide necessary scalability and enables device mobility. While the identities of devices remain the same, their location can change. Contrary to other solutions to mobility, the LISP imposes no overhead because of identity and location separation. The mobile device has the fixed endpoint identifier and using LISP’s dynamic mapping mechanism its route locator can be found. Since this locator can be associated with the network graph, the traditional routing is sufficient to reach the mobile device in its actual location. The key LISP capability is performing efficient mapping of endpoint identifier to locator(s). The presented thesis performs a detailed analysis of mapping mechanisms and proposes operational improvements in Chapter 4.

RINA is not just a contribution to the current architecture but a continuation of the original concept of internetworking. RINA is based on the simple nearly 50 year old observation that every data

¹ **Node** or **Device**: With reference of this thesis it is any equipment connected to Internet capable of communication. E.g. routers, switches, computers, etc.

² Routing Research Group (RRG). For more, please visit website of this former ad hoc group <https://trac.tools.ietf.org/group/irtf/trac/wiki/RoutingResearchGroup>.

³ Internet Architecture Board (IAB). For more, please visit <https://www.iab.org/>.

transfer is interprocess communication between two application processes and as such it requires only a couple of primitive operations. Contrary to LISP, the RINA is less mature. Thus, more research is needed to address the open issues and to demonstrate that RINA can provide a solution to the current obstacles of the Internet. This thesis deals with RINA in Chapter 5. First, the detailed explanation of RINA concepts is presented. The main contribution lies in the formalization of these concepts and definition of simulation models that provide an environment for analysis of various scenarios. By using simulation, properties of the RINA can be evaluated in different scenarios.

The encompassing (and challenging) dissertation goal is to define general naming and addressing theory. Moreover, we want to investigate properties of this theory regarding the impact on the routing. Because nothing impacts routing more (in either positive or negative way) than how names and addresses are deployed to network objects. The underlying goal of this dissertation is to provide a detailed technical overview and analysis of two technologies (LISP and RINA) aimed at improving the current problems of the Internet. The contribution lies in enhancing LISP cache management algorithm and related data transfer to improve its performance. Moreover, we verified LISP contribution functionality with own accurate simulation models. For RINA, fundamental concepts were formalized using finite-state machine diagrams and a comprehensive set of simulation models was developed. Besides these two main achievements, this thesis provides a broad review of the building blocks of internetworking with the focus on naming and addressing concepts. The aim of the thesis is to shed more lights on the fundamental problems of the current Internet architecture and to evaluate two possible solutions.

The thesis is divided into the following chapters. Chapter 2 provides an overview of thesis topics and describes the common theory behind our research. It also introduces current weaknesses of the Internet and describes factors that influence them. Chapter 3 compares proposed or existing solutions. Chapter 4 presents the LISP protocol (its implementation in simulator environment) and covers proposed control plane improvements together with the impact of this proposals on the overall operation. Chapter 5 delineates RINA and its approach towards the system of recursive encapsulation of one general layer, and then it focuses on its globally first simulator implementation and measured security aspects. Chapter 6 draws conclusions from the research outcomes.

2 Networking Fundamentals

- ☞ –“You realize that our mistrust of the future makes it hard to give up the past.” Chuck Palahniuk
- ☞ *From which parts does Internet architecture consists of?*
- ☞ *What problems are tormenting the Internet now?*
- ☞ *How these problems affect current TCP/IP routing concept?*

If we want to be thorough when describing theoretical fundamentals for this thesis, we need to start with the high-level overview of networking and work down to low-level parts. We will start with the fundamental question. Is there an Internet Architecture? In a search for the answer, we must first establish the common dictionary how to understand the word *architecture*. When network administrators and computer network researchers speak about *architecture*, they often use this term to create “nobler” context to things like network technology or network protocol. Let us correct this meaning now.

In this document, we borrow definition used by John Day. **Architecture** is a set of rules and constraints that characterize a particular style of construction. The architecture is a style of construction rather than the construction itself. If we speak about some architecture, we refer to a set of general rules and high-level concepts constituting the architecture. To illustrate this fact, it is just like a making distinction between gothic architecture and a house built in a style of gothic architecture. The great example of proper usage is in ISO/IEC 7498-1 [1] describing the OSI Reference Model (OSI-RM) – Section 5 describes the architecture (“construction style”); Sections 6 and 7 describe a concrete implementation of this architecture using seven layers (“examples of constructions following style”).

The Internet as technology is a continuing sequence of evolutionary steps. However, since its beginning it is all about a few fundamental principles that had not changed. Internet architecture is about best-effort communication with global connectivity across the simple but resilient network where intelligence is on an end-to-end basis rather than hidden in the network as RFC 1958 [2] stated.

The goal of this chapter is to layout computer networking foundations and associate them with more general communication principles. Moreover, we would like to point out present problems of the Internet and discuss their impact on routing table size and control plane load. Now let us use the formal approach to describe foundations of a computer network (in Subchapter 2.1) communication followed by an observation about nowadays problems of the current Internet architecture and their impact (Subchapters 2.2, 2.3).

2.1 Basic Terminology

This subchapter seems to be filled with an exhaustive number of terms and their definitions. However, the aim is to get the reader familiar with all cornerstones of computer systems interactions. Moreover, the goal is to show an association between other cornerstones and to point out their context in the frame of the big picture, which is a *network architecture*. The content of this subchapter is loosely based on [1] and Chapter 2 of the book [3].

Devices **communicate** in order to share state and exchange data in the frame of applications in computer networks. Devices communicate using shared schemes known as protocols.

The **protocol** is a set of prescriptions and procedures that each device participating in communication must follow. Devices utilize protocols to exchange finite quanta of information in the form of **protocol data unit (PDU)**.

Any protocol could be formally defined by a finite-state machine (FSM) or using temporal logic, implementation of this formalism is called **protocol machine (PM)**. PDUs consist of **protocol control information (PCI)** and user-data – PM interprets PCI, the data part is relayed above in the hierarchy.

PMs might be assembled to create layered hierarchy so that the output of one PM is the input of another PM. In this case, a position of the protocol (or its PM) in the hierarchy is denoted by protocol (or PM) **rank**. Let (N)-* be an element * with the rank N then (N+1)-* is an element * one rank above and (N-1)-* is an element * one rank below.

Subsystem is (compound) (N)-element which interacts directly only with (N+1)-elements and (N-1)-elements within (N)-layer. The **layer** is a collection of (N)-subsystems of the same rank N. (N)-service is a capability of the (N)-layer and the layers beneath. Each layer has a **scope** – limit (or boundary) of operation within which layer entities can communicate directly without the help of other layer elements. Subsystem contains active (N)-entities embodying a set of (N)-services, which might include more than just (N)-PM. (N)-function is a part of the activity of (N)-entities.

PDUs might be encapsulated one into another to reflect the hierarchy of PMs in control. **Service data unit (SDU)** is (N+1)-PDU provided to (N)-PM by (N+1)-PM before PCI is prepended to it. Therefore, SDU (whole or part) is placed into (N)-PDU's user-data from the perspective of (N)-PM.

We define following kinds of communication according to coupling of shared state:

- **Association** = minimal shared state without coupling between communicating nodes;
- **Flow** = shared state without tightly coupled elements, often represented by protocols using two-way handshake (without feedback between communicating parties);
- **Connection** = shared state with tightly coupled elements, often represented by protocols using three-way handshake (with feedback between communicating parties)
- **Binding** = fully shared state, often represented by applications with shared memory.

Any sender or receiver passes through following phases in order to communicate. Each phase consists of operations and their inverses, but it does not necessarily imply that PDUs are exchanged:

- 1) *The Enrollment Phase* – Objects of communication are prepared (devices initialized, resources allocated) for a network during this phase. Enrollment includes creation, distribution, maintenance and deletion of information mandatory to make an instance of communication. Addressing information are stored in appropriate directories (i.e., address pool, routing table) and policies are selected. Often this phase covers manual configuration of objects. However, some operations might be automated (e.g., DHCP⁴, SLAAC⁵);
- 2) *The Establishment Phase* – The shared state, necessary for (N)-PMs to communicate, is established in this phase. According to the degree of coupling, associations/bindings between (N)-PM and (N±1)-PM are created or initially shared state for flow/communication of (N)-PMs is synchronized. If **Quality of Service (QoS)**⁶ resources were not allocated during enrollment phase then they are set here (e.g., RSVP⁷);
- 3) *The Data Transfer Phase* – First sent user-data initiates this phase during which (N+1)-PM exchanges SDUs. This phase includes operations necessary to provide the actual transfer of user-data and functions that support it.

The mechanism is a part of the protocol that is fixed and cannot be changed. On the other hand, **the policy** is a part of the protocol that could be deterministically negotiated usually during the establishment phase (e.g., which CRC⁸ polynomial to use for data corruption detection). Tables Tab. 1, Tab. 2 and Tab. 3 sum up basic communication mechanisms, which we categorized into following groups: a) related to *establishment phase*; b) related to data transfer; c) related to data transfer control.

Name	Description
Authentication	Authentication mechanism determines the identity of sender or receiver.
Access Control	Access control mechanism is used to determine whether the requestor is allowed to use a resource or not after successful authentication. Both mechanisms are using a variety of different protocols with flexible policies (e.g., IKE ⁹).

Tab. 1: Mechanisms related to enrollment phase

⁴ Dynamic Host Control Protocol (DHCP). For more, see RFC 2131.

⁵ IPv6 Stateless Address Autoconfiguration (SLAAC). For more, see RFC 4862.

⁶ Quality of Service (QoS): QoS is the overall performance of a telephony or computer network quantitatively measured using such as like error rates, bit rate, throughput, transmission delay, availability, jitter, etc. For more, see https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.800-200809-I!!PDF-E&type=items.

⁷ Resource Reservation Protocol (RSVP). For more, see RFC 2205.

⁸ Cyclic Redundancy Check (CRC) is error detecting code used commonly in computer networks. For more, see http://en.wikipedia.org/wiki/Cyclic_redundancy_check

⁹ Internet Key Exchange Protocol (IKE). For more, see RFC 7296

Name	Description
Addressing	The protocol must have fields how to specify source and destination of PDU in multi-access network segments (e.g., shared half-duplex Ethernet LANs ¹⁰). The address must be long enough to provide unambiguity for all communicating devices. On point-to-point interconnections, (e.g., HDLC ¹¹ , PPP ¹²) address fields in protocols are unnecessary because only two devices are present on the link.
Flow/Connection Identifier	Any protocol supporting multiple instances of communication between the same devices must differentiate between flows/connections in order to deliver PDU to appropriate instance of PM. For this reason, flow/connection-id is included as a part of PCI (e.g., pair of source and destination port numbers in UDP/TCP). Flow/connection-id must be unambiguous within the set of the same rank protocols.
Delimiting	<p>Delimiting is generally packaging of SDU into User-Data fields so it can be re-constructed. Delimiting operation may be comprised of following (N)-functions:</p> <ul style="list-style-type: none"> • One must be able to determine borders of SDUs. Either external or internal delimiter is used. In the case of external, the special pattern is utilized to mark the end of one PDU and beginning of another. The mechanism must guarantee that this special pattern (usually in the form of unique bit sequence) does not occur anywhere inside PDU (i.e., bit stuffing¹³). In the case of internal delimiter, length (count of bits, bytes or octets) of inner PDU is explicitly given as a PCI field of outer PDU. • Data-link technologies often pose some limits on acceptable PDU size. Hence, mechanisms for sender's fragmentation (splitting single (N)-SDU to multiple (N)-PDUs) followed by receiver's reassembling (putting pieces back together) are needed. For IP, a total number of fragments and each fragment-id are stored in separate PCI's fields. <p>For the sake of efficiency, some protocols might combine several SDUs into single PDU (e.g., YMSG¹⁴) or segment one bigger SDU into multiple smaller PDUs. Combination/Segmentation happen between (N)-SDUs and (N)-PDUs of the same rank. Hence, fewer PCI fields are needed when comparing with fragmentation/reassembling.</p>
Ordering	Some protocols need that PDUs are delivered to the receiver in the same order as they were generated by the sender. <i>Sequence numbers</i> in PCI fields achieve this goal. However, use of ordering does not imply that all PDUs are always delivered. The trick is to recognize properly that some PDUs are missing (some of them could be retransmitted; some of them could be correctly discarded).
Relaying	Relaying occurs whenever PDU needs to be passed from one PM to another. It could be provided by (N)-PM and (N+1)-PM whenever device's (N+1)-PM is the receiver of a given PDU. Otherwise, PCI's addresses are used (a process known as routing) to determine (N-1)-PM to which PDU is subsequently <i>relayed</i> in order to get PDU closer to the receiver (a process also known as forwarding).
Multiplexing	The multiplexing is a mapping of multiple (N)-PMs communications (either flows or connections) onto fewer (N-1)-PM communication.
Keepalives	Communication over a longer period without any traffic needs this mechanism to determine that corresponding device is still operational (e.g., routing protocols, IPsec ¹⁵ , peer-to-peer applications).

¹⁰ Local Area Network (LAN). For more, see https://en.wikipedia.org/wiki/Local_area_network.

¹¹ High-Level Data Link Control (HDLC). For more, see ISO/IEC 13239.

¹² Point-to-point Protocol (PPP). For more, see RFC 1661.

¹³ Bit stuffing: Insertion of non-information bits into data in order to protect some special bit patterns. For more, see http://en.wikipedia.org/wiki/Bit_stuffing.

¹⁴ Yahoo Messenger Protocol (YMSG). For more, see http://en.wikipedia.org/wiki/Yahoo!_Messenger_Protocol

¹⁵ IP Security (IPsec) is set of tools and protocols for establishing confidential communication across IP network. For more, see RFC 6071.

Data Anti-Corruption	<p>PDU's transferred over any unreliable medium might experience data corruption (for instance electromagnetic interference or signal attenuation among others). Error detection can detect any bit/byte error using checksum or CRC. Error correction is able not only to detect bit/byte error but also in some cases correct corrupted data (e.g., Viterbi algorithm¹⁶).</p> <p>The integrity protects communication from unauthorized manipulation, i.e., insertion, deletion and alteration of PDU (e.g., variety of one-way hash functions used for computing HMAC¹⁷).</p> <p>Both <i>error detection/correction</i> and <i>integrity</i> are parts of the unifying mechanism called data anti-corruption offering service, which prevents user-data from corruption.</p>
Compression	Policy for this mechanism chooses (if any) available compression algorithm that could be used to reduce the size of PCI of certain protocols (e.g., RTP ¹⁸ header compression where whole RTP stream uses same PCI information).
Confidentiality	Confidentiality of communication means that nobody else can understand PDU's user-data except receiver and sender (e.g., variety of cryptographic algorithms such as DES, 3DES ¹⁹ , AES ²⁰ , RSA ²¹).
Non-repudiation	The non-repudiation mechanism guarantees that all devices of particular communication cannot deny processing of relevant PDU's.

Tab. 2: Mechanisms related to data transfer

Name	Description
Initial State Synchronization	<p>Any shared state between devices must be initialized first. Following forms of initial synchronization are recognized based on [4]:</p> <ul style="list-style-type: none"> • Synchronization representing local association between PMs of adjacent ranks, no PDU's are exchanged, and minimal shared state is required (e.g., UDP); • Synchronization for flow communication utilizing request/response PDU's, used by protocols without any feedback; • Synchronization for connection communication utilizing request/response and acknowledgment PDU's. This operation is restricted by time constraints – TCP changes synchronization state explicitly using flags (e.g., SYN, FIN); delta-t [5] protocol utilizes mainly timer-based mechanism bounding maximum PDU lifetime, the maximum receiver-waiting period before acknowledging PDU and maximum duration that sender tries to resend PDU.
Loss and Duplicity Detection, Retransmission Control, Acknowledgement	<p>Transmission of data over the Internet is by its nature unreliable. PDU's might be lost or duplicated. PCI's numbers are utilized for lost and duplicity detection mechanism (i.e., remaining a gap in the sequence indicates the loss; multiple PDU's with the same sequence numbers indicate duplicity). Duplicated PDU's are discarded; lost PDU's may be retransmitted.</p> <p>The previous technique cooperates with the acknowledgment (ack) mechanism, which is used by the receiver to inform the sender about PDU's that has been received without any problem. Also, the sequence numbers are used inside ack to inform which PDU was the last received by the receiver. Nevertheless, there</p>

¹⁶ The Viterbi algorithm is used to detect and correct bit errors in data streams. More can be discovered at http://en.wikipedia.org/wiki/Viterbi_algorithm

¹⁷ Keyed-hash Message Authentication Code (HMAC). For more, see <http://en.wikipedia.org/wiki/HMAC>

¹⁸ Real-time Transfer Protocol (RTP). For more, see RFC 1889.

¹⁹ Data Encryption Standard (DES) and Triple DES (3DES). For more, see <http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf>

²⁰ Advanced Encryption Standard (AES). For more, see <http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>

²¹ Rivest, Shamir, Adleman (RSA) asymmetric algorithm. <http://www.google.com/patents/US4405829>

	<p>is separate PCI field for sequence numbers referring to lost/duplicity detection and another PCI field for acknowledgment purposes.</p> <p>If an ack is not received in due time, the sender employs retransmission mechanism to generate missing PDUs. If an ack is received, the sender usually deletes PDUs that are pending for retransmission.</p>
Flow Control	<p>This mechanism of flow control prevents sender(s) to overwhelm receiver(s) with data, so it is unable to process them in due time. Flow control is the binary relation between single sender and receiver. Two forms of flow control are known:</p> <ul style="list-style-type: none"> • the credit scheme – receiver tells sender what amount of credit (usually in octets or number of PDUs) it has to send data before getting the new quantum of credit (e.g., the size of TCP sliding window); • the pacing scheme – receiver tells sender how fast (usually at which bit rate) it can send data (e.g., FIR, leaky bucket²²).
Congestion Control	<p>Congestion control [6] mechanism tries to protect network from experiencing congestion collapse (see [7], [8]) – period of low throughput, packet loss and transmission latency. Congestion control is n-ary relation between subsystems. Countermeasures against congestion collapse include:</p> <ul style="list-style-type: none"> • congestion avoidance built into transport protocol (e.g., TCP, DCCP²³); • active queue management (including differentiate services and dropping algorithms like RED²⁴) and congestion notification techniques (e.g., ECN²⁵).

Tab. 3: Mechanisms related to control of data transfer

Complete network architecture should contain hooks for all previously described mechanisms even if the implicit policy for particular mechanism does nothing (e.g., support confidentiality but not applying encryption to outgoing traffic). Tables outline the core set of policies. Nevertheless, we can assemble them together to create more complex mechanisms (e.g., combine confidentiality, integrity, and the random nonce to guarantee anti-replay protection mechanism). Hence, the complete set of mechanisms is practically unlimited. Based on [9], we developed an ontology of data transfer mechanisms, which is depicted in Fig. 1.

Above mentioned terminology and mechanisms provide the framework for Internet operability and foundations for a cooperation of different technologies.

²² Leaky bucket: Algorithm that allows policing or shaping of data traffic to conform some bandwidth or speed restrictions. For more, see http://en.wikipedia.org/wiki/Leaky_bucket

²³ Datagram Congestion Control Protocol (DCCP). For more, see RFC 4340.

²⁴ Random Early Detection (RED). For more, see https://en.wikipedia.org/wiki/Random_early_detection.

²⁵ Explicit Congestion Notification (ECN). Visit https://en.wikipedia.org/wiki/Explicit_Congestion_Notification

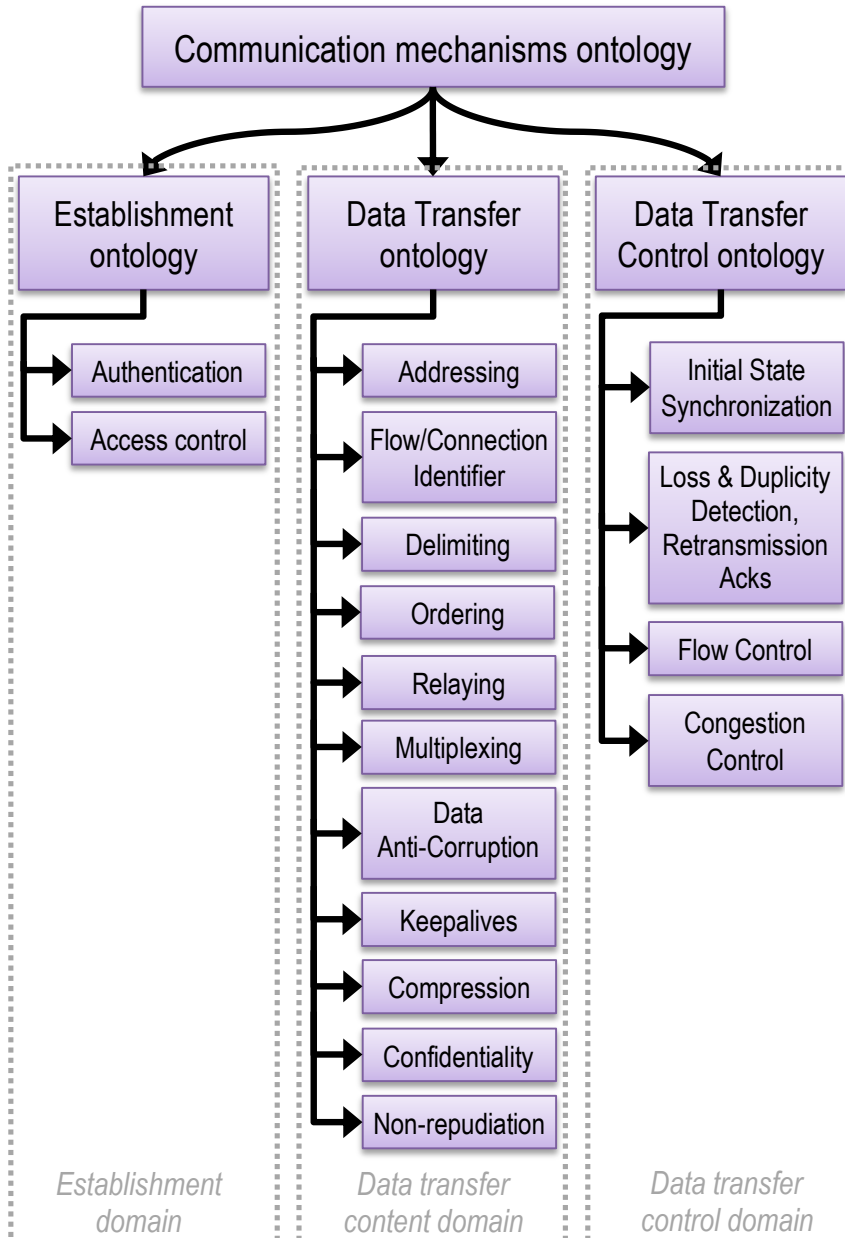


Fig. 1: Data transfer mechanisms ontology

2.2 Present Problems of Internet

Among some driving factors of today's Internet [10] are:

- the widespread availability of wireless (including Wi-Fi and cellular networks) connectivity allowing more non-PC devices perform ad hoc connections;
- deployment of virtualization increasing the number of logical computing systems;
- more cloud computing and peer-to-peer applications changing traffic characteristics towards less deterministic and stochastic models of CDNs²⁶;
- reaching the Zettabyte era more quickly due to the overall increase in broadband speeds.

Issues below are only consequences of Internet usage, which are completely different comparing to Internet conventions and user base 30 years ago.

What we are experiencing is that more and more **hosts**²⁷ and **routers**²⁸ are connected to the Internet every day using different wired and/or wireless technologies. Also growing the amount of transferred data comes hand in hand with an increasing number of users. Paths between nodes on the Internet are becoming shorter, faster, more redundant and more reliable. This trend significantly affect the growth of router table sizes than ever before (observe for instance in chronological order [11], [12]). More existing IPv4 addresses are used as **Provider Independent (PI)**²⁹ rather than **Provider Aggregatable (PA)**³⁰ addresses of Internet Service Provider (ISP)³¹. The free IPv4 address space is depleted, and IPv6 is still fighting to reach at least 5% of overall traffic (see [13] as a representative statistic example of mid-size NREN³²) despite the fact that it has been more than 17 years since its standardization.

Over the past several years, many discussions were held (for more general information, please see [14], [15] and [16]) whether current Internet architecture could sustain its expansion in the middle and long range future. Somebody argues that new better resources are being invented faster than available technology could keep up with them. Somebody disagree that every resource has physical boundaries that cannot be passed on, and that poses as a limiting factor. Nevertheless, the impact of the

²⁶ Content delivery network (CDN). For more, see https://en.wikipedia.org/wiki/Content_delivery_network.

²⁷ **Host**: Device that can send/receive packets, but does not participate in forwarding of packets.

²⁸ **Router**: Device that forwards packets across the network layer.

²⁹ **Provider Independent (PI)** addresses: Address prefix that organization receives from its Regional Internet Registry (RIR). Benefits of using PI addresses relies in fact that if organization needs to change ISP then it does not need to renumber its address space. ISP change means just slight change of routing information propagated to to DFZ.

³⁰ **Provider Aggregatable (PA)** addresses: Address prefix that organization receives from its provider. The PA address advantage is that all networks of a given ISP – components of ISP's address space – could be replaced with single aggregate prefix propagated to DFZ.

³¹ Numbering of Internet is govern by Internet Corporation for Assigned Names and Numbers (ICANN) organization which assigns available prefixes to RIRs. RIR delegates prefixes to Local Internet Registries (LIR) which carry out assignments of address to their customers. LIRs usually operate as ISPs in that area.

³² National research and education network (NREN). For more, see <https://en.wikipedia.org/wiki/NREN>.

current situation on routing on the Internet is something that we can clearly observe and at least partially predict future tendencies even though we have not yet reached limits of nowadays resources.

The most severe and apparent symptoms of broken Internet architecture – namely routing table growth, lack of locator/identifier semantics split, cumbersome multihoming and mobility, ineffective inbound traffic engineering and renumbering due to the change of ISP – are listed down below in Sections from 2.2.1 to 2.2.6. Some of these issues explanations are based on a review from RFC 6227 [17], RFC 4984 [18], some of them from respective community observations of current trends. These symptoms are currently being solved by band-aid mechanisms and architecture patches (e.g., mobility frameworks). However, those solutions usually lack wide-spread deployment to be really deal breakers and/or do not seem to be long-term scalable.

2.2.1 Routing Scalability

The most affected nodes struggling with the situation are **Default Free Zone (DFZ)**³³ routers. Every year the size of **Routing Information Base (RIB)**³⁴ and **Forwarding Information Base (FIB)**³⁵ of those routers increases. The rate, at which prefix count is growing in the RIB, is the object of discussions [19] but it seems to be slightly faster than linear (sometimes called *superlinear*) for a couple of last years [20], [21]. We can see historical progress in the size of Border Gateway Protocol (BGP) [22] RIB and FIB for IPv4 and also IPv6 on the following graphs depicted in Fig. 2, Fig. 3, Fig. 4 and Fig. 5 from [23]. The year is on the X-axis, and the number of prefixes is on the Y-axis.

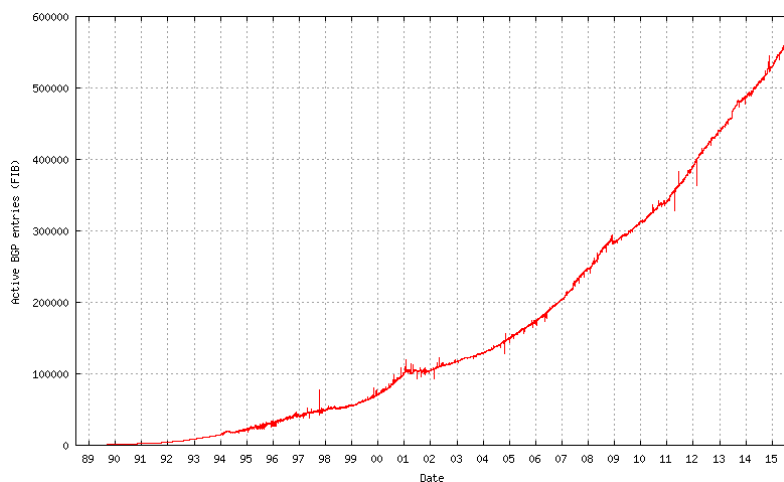


Fig. 2: IPv4 – All BGP entries in FIB

³³ **Default Free Zone (DFZ)**: Backbone of the Internet where routers must keep complete routing tables with all reachable destination networks. In opposite of this are Tier 3 ISP or networks or end customers that are using usually only partial routing information – they have complete knowledge about local connectivity and any other network beyond is available via default route.

³⁴ **Routing Information Base (RIB)**: Basically abstract data structure holding information from a given routing source that holds information about all reachable destination networks and paths to those destinations.

³⁵ **Forwarding Information Base (FIB)**: The FIB is optimized version of RIB. It is consulted most of the time when forwarding packets because it is supported by specialized HW.

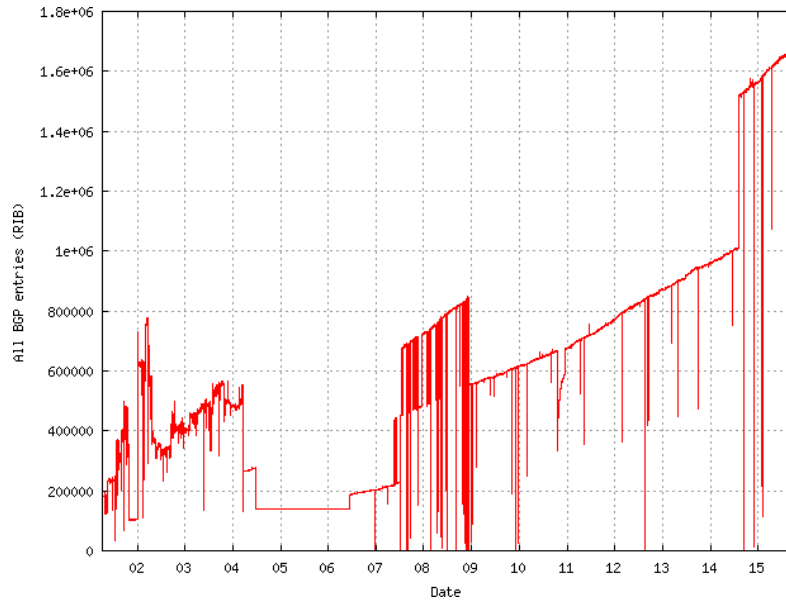


Fig. 3: IPv4 – Active BGP entries in RIB

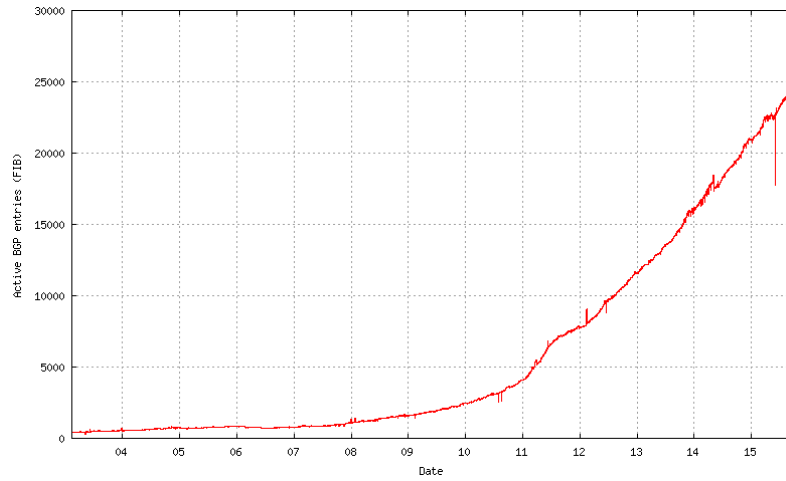


Fig. 4: IPv6 – All BGP entries in FIB

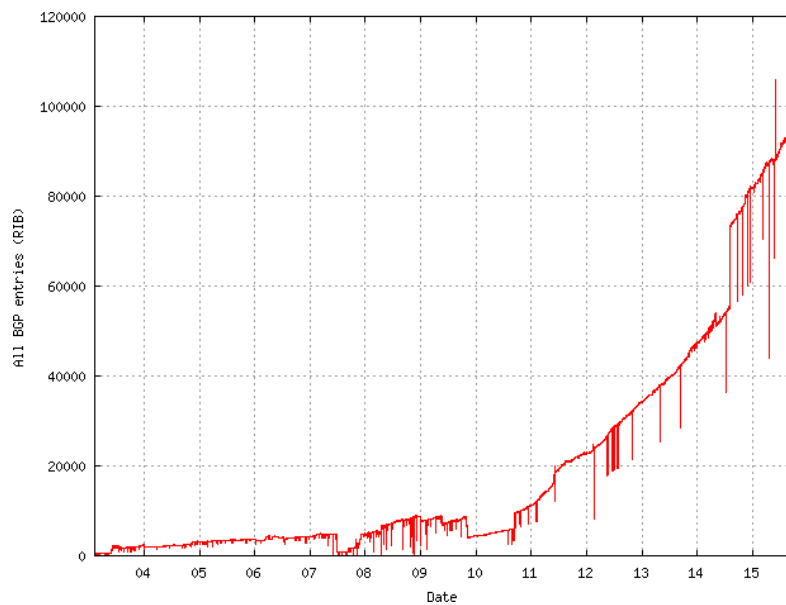


Fig. 5: IPv6 – Active BGP entries in RIB

Current numbers are taken from a router in one of the APNIC research and development **autonomous systems (AS)**³⁶. They are relevant to the date of this publication:

- IPv4 RIB = 1 682 113 prefixes;
- IPv4 FIB = 573 400 prefixes;
- IPv6 RIB = 96 409 prefixes;
- IPv6 FIB = 24 857 prefixes.

Previous numbers mean that this particular router sees 573 400 IPv4 destination networks in today's Internet where there are 1 682 113 different paths to them and vice versa for IPv6. The prefixes count is going to increase with advancing depletion of IPv4 space and progressing deployment of IPv6.

Each prefix must be processed which increases the **control plane**³⁷ load. This raises consumption of router's CPU performance and memory and last but not least increases the size and a potential number of exchanged routing updates. This presents **routing scalability** issue for future routers – sometimes the same problem is also known as *DFZ RIB/FIB growth*.

It is believed [18] (assuming nowadays growth and available hardware and software) that we will still have resources to build devices capable of dealing with this problem efficiently. However, what is becoming a concern is a price of these devices.

The negative consequence can be passing the unreasonable cost of ISP's investments to customers. Potentially, only large Tier 1 ISPs could afford such devices/investments which would also affect maintenance and operation expenses of DFZ routing. Technologically the routers: a) must maintain increasing state information in RIB and converge usable routes quickly enough; b) must populate FIB from RIB fast and must be prepared for enlarging the size of FIB itself; c) must perform forwarding lookups (and at best also routing decisions) at line-rate speeds; d) must use HW that does have reasonable power consumption or cooling demands. From the business perspective, we must understand that DFZ is run solely by private entities without any centralized supervision [24]. They are making a profit from it, so all the policies (like acceptance and processing of prefix) are in their hands. From their perspective, it is not beneficial that cost of routing infrastructure would grow too rapidly especially due to the factors they are unable to control (e.g. increasing number of Tier 3 ISP customers that want to multihome). To put it simpler – no ISP (especially the one operating at DFZ) would be happy from upgrading its infrastructure (buying better routers) to maintain the same level of service quality and availability just because of the growing requirements of the routing system.

³⁶ **Autonomous System (AS)**: Set of devices under one administration domain.

³⁷ **Control plane**: Part of the router that acts as the brain responsible for maintaining various state information such as routing table with the help of routing protocols and handling L3 issues (defragmentation, filtering, traffic classification/marketing, QoS policing/shaping, cryptographic operations, etc.).

The final verdict (see [25]) is that vendors and ISPs are (under current conditions) able to deal with the growth of the Internet. However, scalable and cheaper solution would be welcomed to reduce costs and prepare for future demands.

2.2.2 Decoupling Identification and Location

What is currently called decoupling of location and identification a.k.a. *loc/id split* is merely the result of IPv4 address semantics as described in RFC 2101 [26]. IP address serves multiple roles nowadays:

- 1) *Identification* – **Identifier** is a bit string that is used during the communication’s lifetime. It identifies communicating parties in a way that IP address verifies the source of packets;
- 2) *Localization* – **Locator** is a bit string that specifies packet destination where it should be delivered. It locates the place on the Internet, where a device is attached. Routing protocols interpret IP address as a locator and build up routing tables based on the situation that routers route traffic towards a destination. The locator is also known as **Point of Attachment (PoA)**³⁸.

Identifiers and locators have different requirements on uniqueness and lifetime. Identifiers must be unambiguous on each set of communicating parties while locators must be unambiguous within one or more routing domains. Identifiers must be valid at least during the maximum lifetime of communication between given devices. Locators must be valid as long as a routing system within a routing domain needs them.

Let us focus on real-life implications of the fact that IP address is used both as identifier and locator. What if any node has more than one IP address, which one identifies it? A device is situated in the network at one place. However, PoA addresses do not express device’s position but networks to which device is connected. Moreover, PoA could have an entirely different location from the perspective of DFZ. Another example is multiple virtual machines on one host system. One approach is that we have a virtual network inside host system. However, in this case, we might run into the problem that we have to use NAT³⁹, or we lack address space. Another approach is that virtual machines share host system address. However, how can we then differentiate between virtual machines from a network perspective?

Those discrepancies were observed by many during the last thirty years (e.g., one of the oldest notes about it is in [27]). *IP address overloading* with both previously mentioned functions is one of the major factors causing routing system inscalability [28]. Yakov Rekhter stated so called **Rekhter’s Law** targeting this contradiction:

— “*Addressing can follow topology or topology can follow addressing. Choose one!*”

³⁸ **Point of Attachment (PoA)**: Device’s interface (and address of this interface) by which it is connected to some network reachable via Internet. Device could have and use simultaneously more than one PoA for communication. We will see later in Chapter 3 that this perceiving of PoA based on IETF’s view is flawed.

³⁹ Network Address Translation (NAT). For more, see RFC 1631.

However, it is hard to be in compliance with Rekhter's Law because usually identifiers are assigned based on customer's policy, not topological structure. Hence, the single address space can hardly serve both IP address functions efficiently. Thus, solving wrong IP address semantics dichotomy seems like a necessary thing to do.

When taking into account current TCP/IP status quo, loc/id split would be the natural solution for some problems discussed in this subchapter. The most notable advantages (see [29] for details) of decoupling locator and identifier are: a) reduction of DFZ routing tables because they would contain only locators, which would improve scalability of control plane; b) be design support for mobility and multihoming by employing mapping between two distinct namespaces (to one identifier may belong multiple locators) comparing to hacks when using only single namespace of blurred locators and identifiers. Nevertheless, we are going to show that core of the problem lies elsewhere.

2.2.3 Multihoming

Internet's *multihoming* stands for the situation when the customer is using two or more ISPs for transit services as it is defined in RFC 4116 [30]. Nevertheless, this definition may be limited in application, because it refers only about *multihoming* between autonomous systems. We propose wider definition of **multihoming**, which covers following use-cases:

- 1) multihoming of single host attached redundantly to one or more networks;
- 2) multihoming of single (LAN) network (containing a set of hosts) interconnected redundantly with one or more networks;
- 3) multihoming of autonomous systems (containing a set of networks) interconnected redundantly with one or more ISPs;

Will focus only to multihoming of network(s) (points 1) and 2) of the previous list) throughout the whole Chapter 2. Below are some of the reasons why customers demand *network multihoming*:

- *Redundancy* – Customers are looking for high availability of their services. Hence, their (both customers and ISPs) networks should be operational at best 99.999% of all the time (this represents approximately 5 minutes of allowed outage during whole year) to meet this constraint. From the perspective of Internet connection, this could be accomplished by having more than one ISP to avoid a single point of failure;
- *Load-balancing* – Traffic could be load-balanced between multiple working links leading into/out from customers AS to avoid congestion or to increase the available communication bandwidth;
- *Traffic Engineering* – Customer wants to influence how traffic is handled beyond default routing behavior, e.g., for example, to avoid problematic paths, to isolate some sets of addresses, etc. (for more, see RFC 2260 [31]);

- *Transport-Layer survivability* – BGP driven multihoming provides at some level (i.e., successful convergence in certain time frame) session survivability for transport protocols.

A mandatory prerequisite for multihoming is that every customer is uniquely identified on the Internet – this is done by **autonomous system number (ASN)**⁴⁰. Multihoming is nowadays accomplished with the help of BGP, which informs others about the path to customer’s network via two or more ISP transit systems.

Multihoming works with PA and PI addresses. For both cases, customer’s prefix is propagated to DFZ. Nevertheless, for PA case only primary ISP (i.e., assignee of customer PA addresses) aggregated prefix is present in DFZ and additional routing table entry appears only during path failures to this primary ISP. A multihoming problem arises when a customer's PA prefixes are advertised by non-primary AS(es). Because of the **longest-prefix match**⁴¹ routing lookup, the customer's traffic will be directed through the non-primary AS(s). The primary ISP is then forced to deaggregate the customer's PA prefix in order to keep the customer's traffic flowing through it instead of the non-primary AS(s).

The trouble with multihoming is closely connected with IP address semantics described in the previous section – IP addresses is a PoA which is route dependent (i.e., reachability of multihomed networks depends on the chosen/available route). However, IP *routing* should be route independent, but this cannot be satisfied when it takes into account destination and next-hop IP addresses which are route-dependent PoAs.

Assume network graph in Fig. 6 with one router connected with two interfaces (two PoAs) to different ISPs for the sake of requested connection redundancy. If one PoA experiences outage (e.g., 192.168.1.1 on primary red route), then it does not imply that router and LANs behind it are unavailable. The routing algorithm can find a backup route for LANs, but it cannot help to reroute PDUs intended for PoA, which is currently down. Multihoming is not inherent use-case to IP. Route dependency of multihomed networks remains unsolved despite the fact that it firstly appeared in 1972 (more than 40 years ago) as Tinker Air Force Base multihoming request [3].

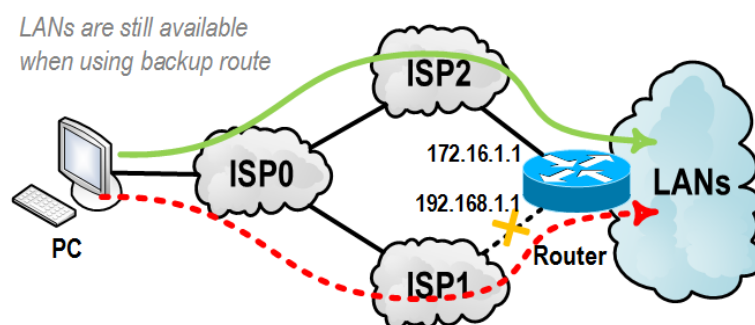


Fig. 6: Network multihoming use-case illustration employing simplified BGP rerouting

⁴⁰ **Autonomous System Number (ASN)**: Globally unique identifier 16 or 32 bits long assigned by ICANN and maintained in online database on <http://www.iana.org/assignments/as-numbers/as-numbers.xhtml>.

⁴¹ **Longest-prefix match**: Algorithm used by routers to retrieve the best available (the most accurate) entry from routing/forwarding table or any other table containing IP network entries. For more see D.E.Comer, Computer Networks and Internets (5th ed.), p. 368, ISBN 978-0-13-606698-9, 2008.

2.2.4 Mobility

During the last years, the idea of the **Internet of Things (IoT)**⁴² became more real and widely accepted as probable use-case of Internet. Some predictions expect that 20-75 billion nodes will be connected to the Internet by the year 2020 [32]. Basically, a throng of devices with own IPv6 addresses would need access to the Internet. **Mobility** is the ability of a node or whole network to change its topological connectivity without disruption of ongoing communication (remark: *application mobility* is not covered in this thesis though it is often associated with this term). Authors of TCP/IP stack had never thought about this use-case. Thus, IETF had to supplement solutions like Mobile IP [33], Mobile IPv6 [34] or HMIPv6 [35] or Multipath TCP [36] later.

These solutions include:

- a) Dynamic renumbering of mobile entity – considered unsuitable because dynamic IP address change without any further notice may disrupt existing communication;
- b) Renumbering and creating a tunnel between old and new location – it requires the deployment of the *home agent* and *foreign agent* concepts known from cellphone networks;
- c) The ability of a mobile entity to actively announce its new location – usually comes hand to hand with dynamic changes to DFZ routing tables as the mobile entity moves from one location to another.

Portability is another term often discussed with mobility. Portable network address does not change (its format and structure) despite replacing ISPs. All PI addresses are by their nature portable.

A looming current problem (for not just IoT) is how to accommodate possibly billions of smartphones, tablets, printers, and PDAs with the IPv4/IPv6 capability to access the Internet and to provide session survivability when those devices roam from one network to another. NAT is often being used to overcome this limitation by rewriting persistent address to dynamic mobile address. However, NAT breaks **end-to-end principle**⁴³ [37] and due to that NAT is being considered as the temporary fix rather than a solution. Mobility should not be attained feature of some special protocol or technique. Therefore, mobility support should be inherent to the network architecture.

⁴² **Internet of Things (IoT)**: Refers to unique identification of objects in Internet where nearly any device is equipped with IP address and capable of communication via IP. It is merely buzzword overused in marketing expectations of future Internet growth. More at http://en.wikipedia.org/wiki/Internet_of_Things

⁴³ **End-to-end principle**: Application-specific functions ought to reside in the end hosts of a network rather than in intermediary nodes.

2.2.5 Traffic Engineering

Traffic directing and diversion to use other paths than those precomputed by IGP⁴⁴/EGP⁴⁵ is called **traffic engineering (TE)**. We differentiate between two types according to direction of traffic flow:

- *Outbound traffic engineering* – Intra-AS TE, where we try to influence how traffic is leaving AS. IGP metrics is usually altered to support this goal so that preferred exit from AS is utilized. Another way, how to accomplish outbound TE, is to depreferentiate or to filter some routes from BGP neighbors;
- *Inbound traffic engineering* – Inter-AS TE, where more specific routes are propagated with the help of BGP to divert traffic from normal paths (aggregated prefixes). Those altered specific routes are more preferred because they temper BGP decision process [22], [38].

Nowadays inter-AS TE is done rather than intra-AS TE. The reasons to do TE, are similar just as in the case of multihoming. Among those reasons are policing (to restrict transition of certain traffic through a given AS), cost reduction and support of various QoS and **Service-level Agreements (SLA)**⁴⁶.

TE is performed by tuning BGP attributes of the certain routes and/or introducing more specific prefixes into DFZ routing tables. This effectively increases RIB and FIB sizes and presents an additional load to the control the plane. Moreover, network administrators spent hours configuring TE only to discover that the neighboring BGP peer completely rewrites (or ignores) routes attributes, thus preventing the rest of the Internet to learn and conform to intended TE. Hence, network architecture should support nonrefusable TE by design.

2.2.6 Renumbering

Usually, the organization has one ISP where its network is completely inside ISP's AS. In this case, the **organization**⁴⁷ does not need to advertise its network prefix globally because it is a part of provider address space – PA addresses is assigned to the organization. However, if an organization wants to change ISP, then it must be prepared to **renumber** all its nodes according to PA address block enforced by a new ISP. Another option is to ask Regional Internet Registry (RIR)⁴⁸ for PI address block, but there are two drawbacks associated with it:

⁴⁴ Interior Gateway Protocol (IGP). For more, see http://en.wikipedia.org/wiki/Interior_gateway_protocol.

⁴⁵ Exterior Gateway Protocol (EGP). For more, see http://en.wikipedia.org/wiki/Exterior_gateway_protocol.

⁴⁶ **Service-level Agreement (SLA)**: SLA is an agreement between two or more parties, where one is the customer and the others are service providers. SLAs commonly include segments to address: a definition of services, performance measurement, problem management, customer duties, warranties, disaster recovery, and termination of agreement. For more, see https://en.wikipedia.org/wiki/Service-level_agreement.

⁴⁷ **Organization** a.k.a. **Customer**: Entity operating end network with own addressing plan and routing policies.

⁴⁸ **Regional Internet Registry (RIR)**: Organization that manages allocation and registration of internet numbers (IP addresses, autonomous system numbers, well-known port, etc.). Currently world is divided into five RIR based on geographical position: AfriNIC, ARIN, APNIC, LACNIC and RIPE NCC.

- 1) The organization still would not avoid at least initial renumbering when changing from PA to PI addresses.
- 2) The demand could not be met because RIR is already missing PI prefixes large enough (especially with IPv4 address space depletion), or it is against RIRs regulations. PI addresses make the process of migrating between ISP easier; still each PI prefix must be separately advertised to DFZ.

Not only renumbering process could be costly and error-prone (see RFC 5887 [39]) even with the existence of automated tools (e.g. DHCP, SLAAC, etc.), but also some of the organizations may feel stuck or being held as a hostage of their ISPs that provide them with PA prefix.

The renumbering problem grows with the size of the network and number of nodes it contains. Moreover, change of host's addresses negatively affects **access control lists (ACLs)**⁴⁹ and firewall setups or configuration files outside the scope of renumbered network.

⁴⁹ **Access control lists (ACLs)**: A list of permissions attached to an object (e.g., interface, file, service). ACL usually consists of one or more entries (ACE) that are being evaluated whenever object is accessed.

2.3 Influencing Factors

Previously mentioned problems could be non-disjunctively divided into two major groups as those: a) negatively influencing routing table size; b) negatively influencing routing table processing. Except those also some questionable techniques or decisions do exist that are employed by ISPs, and that will be also described in this subchapter.

2.3.1 Burden on Routing Table Size

When speaking about adding pressure to the routing table size the de-aggregation of address prefixes (i.e., *more specific prefixes*) turns out to be a major reason behind DFZ RIB/FIB growth. Among elements why it is happening belong:

- *Traffic Engineering* – Additional TE specific prefixes are advertised;
- *Multihoming* – In case of both PI and PA addresses, the organization's non-aggregatable prefix must be propagated to DFZ so that multihoming could take effect;
- *End-site Renumbering* – Many customers require PI address space to avoid possible renumbering when changing ISP. By its meaning, all PI addresses are part of DFZ that could not be aggregated;
- *Business Acquisitions* – Networking infrastructure could be a constituent part of assets when a company is selling off some of their business. Unfortunately, this could lead to partition of address space to smaller blocks that could not be summarized and these fragments must be advertised separately to DFZ;
- *RIR Allocation Policies and IPv4 Address Exhaustion* – Organizations, are acquiring address space from RIRs. If an organization needs more addresses, then it asks for another block. However, this block is rarely the adjacent one in the address space by the block, which was already assigned to the organization. This leads to assignment of discontinuous prefix blocks to same AS;
- *Dual-Stackness* – Currently IPv6 exists simultaneously together with IPv4 because dual-stack is the only option for non-failure deployment of a new network protocol without the *Flag Day*. Unfortunately, this means that DFZ routers must support coexistence of two routing tables – one for IPv4 and another for IPv6. However, both routing tables are used to find paths to same destinations in case of single AS reachable via IPv4 and also IPv6;
- *Anti-Route Hijacking* – Certain organizations propagate a set of smaller specific prefixes rather than one aggregate. The reason is to avoid potential (or accidental) hijacking of their address space by some other unauthorized party. Unfortunately, this technique only stresses DFZ routing tables more.

Reader’s discretion is advised (the goal is to provide merely proof of existence) when interpreting following (simplified) conclusions drawn from (the current snapshot of) BGP metrics.

What is the size of potentially complete IPv6 FIB? Does IPv6 pose a threat to DFZ routing table sizes and HW of dual-stack routers? Let us quantify the issue of growing size of routing tables based on data provided by [40], which is the snapshot of BGP state on the router in AS 131 072 from 27th September 2015. Tab. 4 contains relevant absolute numbers and discussion follows below:

Description	Parameter	IPv4	IPv6
The total number of prefixes in FIB.	Prefix Count	573 333	24 833
The number of prefixes assigned to organizations by RIRs and LIRs.	Root Prefixes a.k.a. CIDR Aggregates	274 608	16 892
The number of prefixes beyond root prefixes aggregates.	More Specific Prefixes	298 725	7 941
Single AS Path prefixes from the subset of more specific prefixes.	Specifics where AS Path Matches Aggregate	142 878	3 883
The total number of ASes that do exist.	AS Count	51 821	10 251
End site or so-called origin-only AS.	Origin-only ASes	44 282	8 310
ASes that carries only traffic between other ASes, e.g. Tier 1 ISPs.	Transit-only ASes	214	143
ASes that serves both purposes – origin and transit, e.g. Tier 2/3 ISPs.	Mixed ASes	7 325	1 798
ASes that propagate to DFZ only one prefix.	ASes Advertising a Single Prefix	20 544	7 412
ASes that are not multihomed.	Origin ASs Announced via a Single AS Path	33 391	7 633

Tab. 4: Observations about DFZ based on BGP functionality

Some simple yet factual conclusions could be drawn from values in the table. The worst case is a moment (let us call it **saturation point**) when the same IPv4 and IPv6 routing tables would coexist side by side. It is very hard to predict when it will happen. Nevertheless, we can estimate how saturation point would look alike employing now available data.

Let us assume that the lower bound of complete IPv6 routing table is the same as the number of active AS reachable via IPv4 in case that each AS would need only the single aggregated route. That guarantees the smallest yet operational IPv6 routing table:

$$\text{Minimal IPv6 FIB Size} = \underline{51\,821 \text{ prefixes}}$$

Unfortunately, the single prefix for AS cannot allow multihoming or any ingress TE.

Let us look on the multihoming through the following equation catching relationship between $\frac{\text{Origin ASs Announced via a Single AS Path}}{\text{AS Count}}$. The result tells that:

- 64.4 % of IPv4 and 74.5 % of IPv6 ASes do not use multihoming because there is only single path leading to their AS;
- consequently, it could be concluded that 35.6 % of IPv4 and 25.5 % of IPv6 ASes deploy multihoming.

Similarly, employment of traffic engineering could be guessed from $\frac{\text{ASes Advertising a Single Prefix}}{\text{AS Count}}$. The results are:

- approximately 39.6 % of IPv4 and 72.3 % of all IPv6 ASes do not have any TE requirements (otherwise they would be advertising more than one prefix);
- the remaining 60.4 % of IPv4 and 27.7 % of IPv6 ASes utilize more specific prefixes in a manner described above (TE, business mergers, and acquisitions, etc.), or they purposely fragment address space.

As stated above more specific prefixes are mostly used for purposes of multihoming and ingress TE. Hence, let us inspect ratio between $\frac{\text{More Specific Prefixes}}{\text{Prefix Count}}$. Approximately 52.1 % of IPv4 and 32.0 % of all IPv6 prefixes constitute more specific parts. Figures Fig. 7 and Fig. 8 show the trend of ratio in percentiles of more specific prefixes in all advertisements. Trend is stable for IPv4, roughly the half of all prefixes. For IPv6, FIB is experiencing very mild linear increase towards one third of all prefixes.

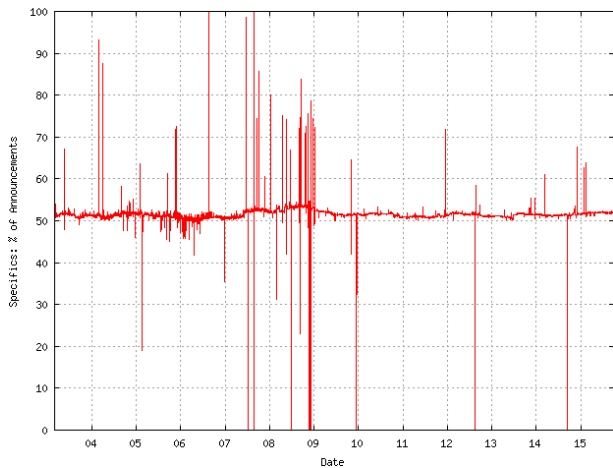


Fig. 7: Percentil of IPv4 more specific prefixes

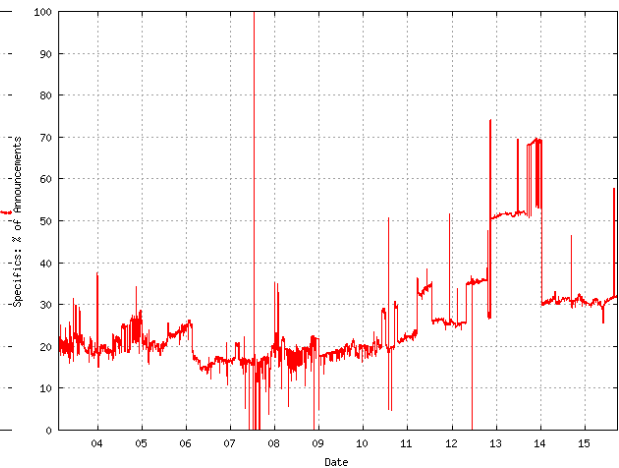


Fig. 8: Percentil of IPv6 more specific prefixes

Now we can extrapolate the estimation of the “real” IPv6 FIB size based on previous data. We expect that each autonomous system needs its aggregate plus all more specific prefixes that are necessary to achieve the same level of multihoming and TE as today. We can use either IPv6 or IPv4 percentile of more specifics copying trend of each address family.

$$\begin{aligned}
 \text{"Real" IPv6 FIB Size} &= \text{Minimal IPv6 FIB Size}(1 - \text{More Specific Prefixes Ratio})^{-1} \\
 &\doteq \begin{cases} 51\,821(1 - 0.32)^{-1} \doteq_{\text{IPv6}} 76\,207 \text{ prefixes} \\ 51\,821(1 - 0.521)^{-1} \doteq_{\text{IPv4}} 108\,186 \text{ prefixes} \end{cases}
 \end{aligned}$$

Let us take into account more conservative input from two results and that is the one using IPv4 percentile. The complete "Real" IPv6 FIB would contain approximately 108 000 records. This size would represent roughly 19 % of nowadays IPv4 FIB. However, IPv6 uses only 24 833 prefixes currently that is just 23 % of outlined "Real" size, which corresponds with the fact that about one fifth of all ASes are already dual-stack. Nevertheless, it is expected that real-life IPv6 routing table would be smaller by number of non-contiguous prefixes assigned to a single organization because of initial larger address space block (prefix length /48 allows a lot of networks for single entity) and because of lack of address space preallocation. Hence, presumably IPv6 FIB size would be somewhere between *Minimal* and "Real" versions.

Hypothetical DFZ router supporting simultaneously perfect dual-stack environment would need to hold at least 681 519 prefixes in its FIB, which is 14 % increase comparing to present overall (IPv4 plus IPv6) FIB size.

Let us draw conclusions from previous calculations. We claim that DFZ routing table size will be still manageable (just 14 % larger than today) even during saturation point. Hence, fears about HW not keeping the tempo with DFZ routing table size growth are based on false premises.

2.3.2 Burden on Routing Table Processing

The previous section discusses factors impacting routing table. Even if we establish a theoretical lower bound on the size of FIB, then we must take into account the amount of control plane work needed to maintain the routing table. The count of routing updates has the major influence on control plane processing delay. Among elements impacting it belongs:

- *Interconnection Richness* – The Internet is becoming flatter in a sense that more and more different paths exist between the same ASes [41]. Increased the number of control plane best route computations is necessary because of that. These computations occur whenever a new route becomes available or also during the change of route attributes. Unfortunately, this interconnection richness is stressing control plane seriously, and it occurs even though the prefix count remains the same;
- *Traffic Engineering* – More specific prefixes with different attributes expressing desired TE effect place more overhead on control plane;
- *Multihoming* – Multihoming AS neighboring with more than one ISP (transit AS) requires more than one interconnection leading towards DFZ. Topology change must be propagated in the form of a routing update whenever a failure occurs. On the contrary, single-homed AS poses no pressure on DFZ control plane load because ISP internally processes any change in connection status;
- *Rapid Shuffling of Prefixes* – Some ASes deploy rapid shuffling of prefixes in order to divert traffic to less loaded links or to optimize traffic by depreferencing (or even canceling) certain

routes that do not meet SLA criteria. Any measurement system, which actively alters routing updates, only increases overall load;

- *Anti-Route Hijacking* – Owning AS advertises purposely more specific prefixes as the countermeasure when fighting against **IP hijacking**⁵⁰. Of course this approach has significant overhead comparing to ideal state when only a single aggregate targeting the same address span is being advertised by AS;
- *Operational Ignorance* – A part of routing updates that are propagated to DFZ appears due to the ignorance of AS network administrators. There might be several reasons for it: a) default behavior of some BGP configurations advertise everything from RIB; b) good aggregation of internal space and optimization of routing updates needs some level of expertise and introduce additional work, which certain network administrators do not have; c) filtering rules (see RFC 7454 [42]) at the borders of AS are not applied which often leads to spoofing of IP prefixes or propagating private addresses.

The previous list outlined some of the reasons, why are there many more specific prefixes in BGP and why is the router’s control plane bothered with occasional routing updates.

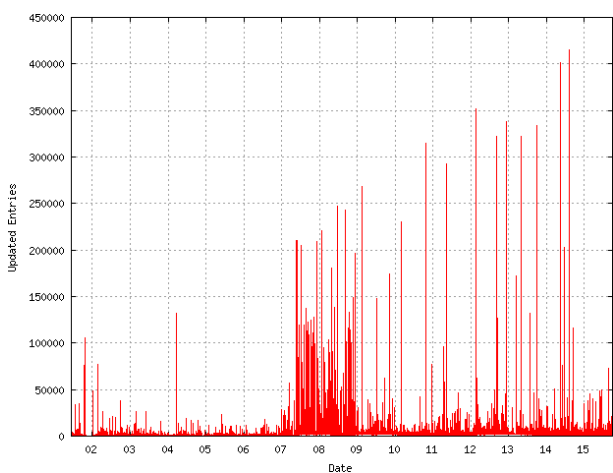


Fig. 9: IPv4 FIB table updates

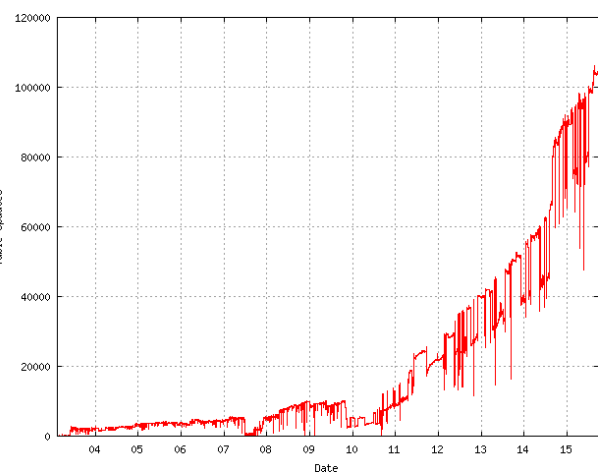


Fig. 10: IPv6 FIB table updates

Graphs in Fig. 9 and Fig. 10 depict the number of FIB table updates for both IPv4 and IPv6. Currently, BGP is experiencing approximately 1 500 updates per hour for IPv4. If there are peaks in IPv4 then they are getting larger and massive (two orders of magnitude) comparing to the usual state. What is more surprising is that this number is approximately 104 400 updates per hour for IPv6. This implies that current IPv6 setup is more intensive on the control plane.

⁵⁰ **IP hijacking**: Illegitimate takeover of groups of IP addresses by corrupting Internet routing tables usually by exploiting BGP functionality. For more, see https://en.wikipedia.org/wiki/IP_hijacking.

2.4 Chapter Summary

In this chapter, we tried to define *network architecture* and its fundamentals by settling on basic terms and definitions. Those basic elements (e.g., connection, layer, scope, rank, protocol-machine, PDU, etc.) are present in current TCP/IP architecture, just as in any other hypothetical architecture comprising computer network communication.

Subsequently, we mentioned problems tormenting nowadays Internet – routing (in)scalability, decoupling location and identification, cumbersome multihoming, overcomplicated mobility, the impact of inbound traffic engineering and unwieldy renumbering of end-networks. We outlined negative consequences in the frame of TCP/IP for each mentioned problem. We tried to express the severity of these issues in numbers focusing on their impact on routing table size and control plane load. We raised some presumption together with hypothetical future trends drawing on publically available global-scale routing data.

This chapter content should support the conclusion that also others come to – the current Internet architecture shows design flaws and sooner or later it will face the crisis emerging from consequences of its poor design.

3 Naming and Addressing Concepts

☞ –“Now you people have names. That's because you don't know who you are. We know who we are, so we don't need names.” Neil Gaiman

☞ *Can we formulate any encompassing theory of naming, addressing and routing?*

☞ *Are there any similar concepts? How does current Internet reflect this theory?*

☞ *What about any solutions dealing with aforementioned problems?*

Problems of addressing and naming are closely connected with networking since its beginning. It directly affects the efficiency of routing and forwarding. Once syntax and semantic of device addressing are employed, the whole system is hard to change. The current Internet addressing scheme is the most obvious example of this problem. Although the present IPv4 address scheme has improved since its definition in the 1980s, it currently represents the major obstacle not only because of address depletion problem. IP protocol designers made multihoming and mobility very difficult and missed a chance to reduce router table size by addressing the interface.

The role of IPv4 is to identify and localize the interfaces of connected devices. However, this assumption poses a great limitation on communication and affected other design concepts. IPv4 protocol address semantics works fine if address assignment follows the network graph and network devices are preserving their membership to local networks. IPv4 communication between network applications requires identifying addresses of network interfaces where the applications are reachable. Enabling IPv4 address change during communication would require modification of datagram delivery mechanism causing complications for network devices as well as for end points. IPv4 routing architecture can efficiently react to connectivity changes detecting dead routes or identifying new routes or routes with better metrics. While exterior gateway routing protocol BGP provides flexibility for propagating information about relocating IPv4 address this always leads to growing global routing tables because of breaking address to topology location dependency. This has a negative impact on routing performance.

The goal of this chapter is to provide the necessary background for the practical part of this dissertation thesis (next two chapters). We try to outline basic motivation why naming and addressing are still issues of current Internet architecture, which is majorly based on Vint Cerf's and Robert Kahn's TCP/IP from 1974.

In the first subchapter, we layout basic terminology using formal apparatus. Next, we discuss other non-computer networking systems, where naming and addressing also occur because we would like to find similarities. In Subchapter 3.3, we try to synthesize working theory employing knowledge from acclaimed articles on this topic. Then, we test compliance with TCP/IP related protocols and tools with this theory. The longest Subchapter 3.5 describes conceptual properties of the ideal solution and introduces many of existing candidates.

3.1 Basic Terminology

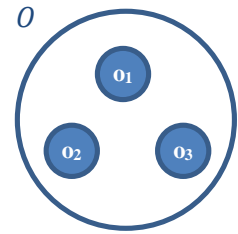
This introduction provides theoretical foundations of naming and addressing. Namely it puts together all related knowledge with profound and utmost respect to papers by John Shoch [43], Carl Sunshine [44], Jerome Saltzer [45], Noel Chiappa [46] and John Day [3].

Natural thinking about basic terms yields following meanings:

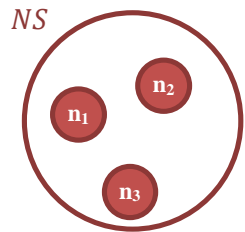
- the object is a structure that is considered to be worthy of the specific name or address;
- the name identifies what the object is;
- the address identifies where the object is;
- the route identifies how to get to the object;

Naming

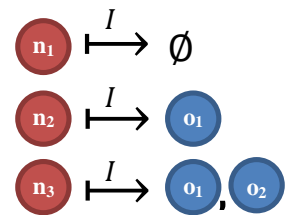
Let us start with an object. **Object** o is a software (or hardware) structure that is considered to be worthy of identification (e.g., variable, service, interface). All *objects* of the same type form a set $O = \{o | o \text{ is object}\}$. We can work with a single *object* or a subset of *objects*, thus it is important to define *power set of objects* $\mathcal{P}(O)$.



Now, let us settle on the meaning of the following terms regarding naming. To be more accurate and consistent within this theory, we define **name** as a *string over the alphabet*⁵¹: $\forall n \text{ is name} \Leftrightarrow n \in \Sigma^*$. However, it is important to note that *name* may be any kind of identifier (e.g., string, color, number). All possible names form the **namespace** as a set of *names* $NS = \{n | n \text{ is name}\}$: $NS \subseteq \Sigma^*$ from which all *names* for a given set of *objects* are taken.



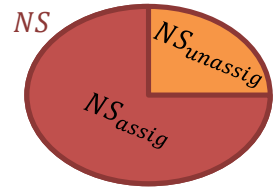
Any *name identifies* (a subset of) *object(s)*, **identify** is relation $I: NS \times \mathcal{P}(O)$. Previous definition allows *name* to *identify* none, one or even more *object(s)* of O . Identifying more than one *object* may be useful for use-cases such as multi-cast or broadcast communication.



Imagine space of IPv4 addresses; some address blocks are assign to owners (e.g., FIT-BUT’s address block 147.229.0.0/16), some addresses from these blocks are being assigned and actively used by devices (e.g., private addresses), some addresses cannot be even sold (e.g., block 240.0.0.0/4 of reserved class E addresses). Naming theory should be granular enough to support all previous use-cases.

⁵¹ Let Σ be alphabet, the set of symbols $\{a\}$. Σ^* is the set of all finite sequences w in alphabet Σ in form $w = a_1 a_2 a_3 \dots a_n$, where any symbol $a_i \in \Sigma$ for $i = 1, \dots, n$. We call w as the **string over alphabet**.

Assignment marks *name* in the *namespace* as available for *binding*, **deassignment** reverses this operation. Hence, the *namespace* is composed of two disjunctive sets of *names*, *assignable* NS_{assign} and *unassigned* $NS_{unassign}$:
 $NS = NS_{assign} \cup NS_{unassign}; NS_{assign} \cap NS_{unassign} = \emptyset.$



Binding is choosing a mapping from *assigned name* to a particular (subsets of) *object(s)* xor (subsets of) *name(s)*; unbinding reverses this operation:

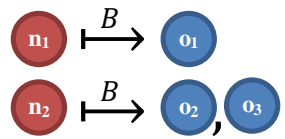
- *binding* is relation $B: NS_{assign} \times M, M = \mathcal{P}(NS) \cup \mathcal{P}(O).$

Name can be either *bound* or *unbound* (available for *binding*):

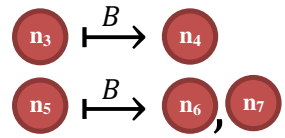
- *name* $n \in NS_{assign}$ is *bound* $\Leftrightarrow \exists m \in M: (n, m) \in B;$
- *name* $n \in NS_{assign}$ is *unbound* $\Leftrightarrow \forall m \in M: (n, m) \notin B.$

Please notice, that *name* can be *bound* to either *object(s)* a.k.a. **direct alias** or other *name(s)* a.k.a. **indirect alias**. Improper *indirect aliasing* may cause circular referencing (e.g., *name* “a” is bound to *name* “b” and *name* “b” is bound to *name* “a”), which is undesired. Hence, a chain of bindings should end with *direct aliasing* providing *identification* of (set of) *object(s)*.

direct aliasing

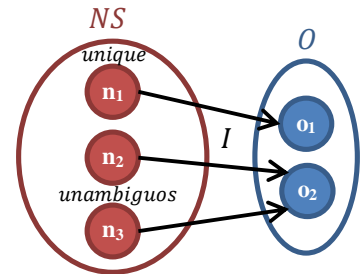


indirect aliasing



We can measure distinctiveness of *name* using following adjectives. **Unique** indicates that there is one and only one *identifying name*, whereas **unambiguous** indicates that there is possibly more than one *identifying name*:

- *name* $n \in NS$, which *identifies* $o \in \mathcal{P}(O)$, is *unique*
 $\Leftrightarrow \exists n, \bar{n} \in NS: (n, o) \in I \wedge (\bar{n}, o) \in I \rightarrow n = \bar{n}.$
- *name* $n \in NS$, which *identifies* $o \in \mathcal{P}(O)$, is *unambiguous*
 $\Leftrightarrow \exists n, \bar{n} \in NS: (n, o) \in I \wedge (\bar{n}, o) \in I.$



Indirect aliases may be *bound* to *unique name* without breaking its uniqueness. Usage of multiple *direct aliases* changes the *unique name* to *unambiguous*.

Making Address Topological

Before investigating terms concerning *address*, we need to define terms related to *topology*, which are based on [47]. **Topology** on a set X is a collection \mathcal{T} of subsets X having following properties:

- \emptyset and X are in \mathcal{T} ;
- The union of the elements of any subcollection of \mathcal{T} is in \mathcal{T} ;
- The intersection of the elements of any finite subcollection of \mathcal{T} is in \mathcal{T} ;

Fig. 11 illustrates three examples of topologies $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ (in compliance with definition) and three examples of non-topologies $\mathcal{T}_4, \mathcal{T}_5, \mathcal{T}_6$ (properties of topology are not met).

Topological space is an ordered pair (X, \mathcal{T}) consisting of a set X and *topology* \mathcal{T} on X .

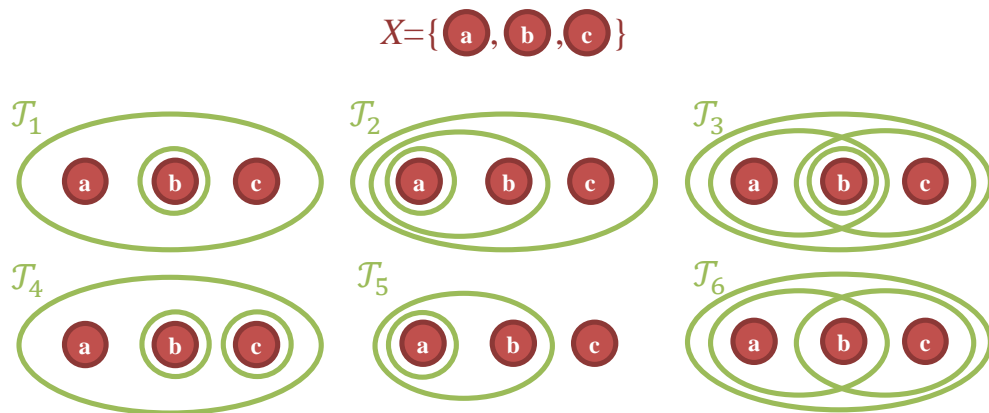


Fig. 11: Examples of topologies and non-topologies

Function $f: X \rightarrow Y$ between two *topological spaces* (X, \mathcal{T}_X) and (Y, \mathcal{T}_Y) is called a **homeomorphism** if it has the following properties:

- f is a bijection (one-to-one and onto);
- f is continuous;
- the inverse function f^{-1} exists (and f is an open mapping);

If *topological spaces* (X, \mathcal{T}_X) and (Y, \mathcal{T}_Y) are *homeomorphic* (if *homeomorphism* exists) then it is guaranteed that points “near” point $x \in X$ are mapped to points “near” point $y \in Y$ (e.g. in Fig. 12);

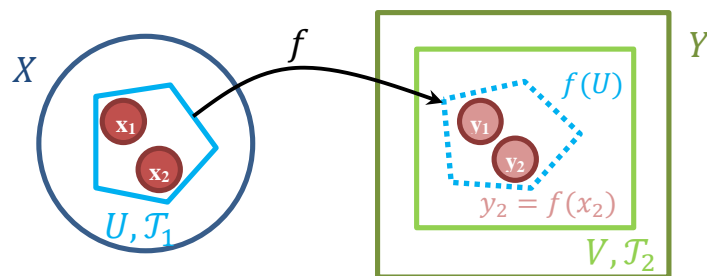


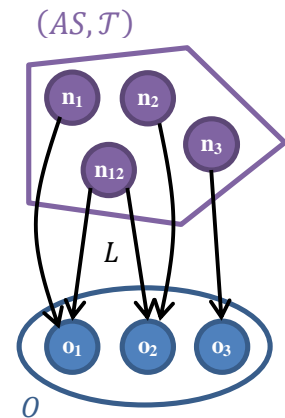
Fig. 12: Homeomorphism illustration

Addressing

Let us return to terms important for addressing. The **address** is a **topologically dependent name** (i.e., *address* contains leads about the position in topology). **Address space** AS is a set of *addresses* $AS = \{a | a \text{ is address}\}$ with a given scope. Address space is topological space, it is a *namespace* with a *topology* \mathcal{T} imposed on it: $(AS, \mathcal{T}): AS \subseteq NS$.

We can perform same operations (e.g., *assign*, *bind*) and observe same properties (e.g., uniqueness) with *addresses* as with *names*. *Address locates* (a subset of) addressable *object(s)*: **locate** is relation $L: AS \times \mathcal{P}(O)$. Instead of *identifying*, we are using term *locating* concerning *addresses*. However, both *identify* and *locate* are relations with same outcomes.

Please note, the address field in a protocol is mechanism but how the address is assigned or its syntax and semantics is policy. Addresses are associated with layers not protocols.



Resulting Properties

The *name* need not to be meaningful throughout the domain and need not be drawn from a uniform *namespace*, whereas the *address* must be meaningful and must be drawn from uniform (flat or hierarchical) *address space*. Flat *address space* has limitations; most notably no hierarchy leaves routing action without any help. Hierarchical *address space* has pros (reduction of routing table sizes) and also cons (what is topologically close may be far away on hierarchical tree branches, which leads to suboptimal routing). However, any structure/hierarchy in the *name* or *address* is intended to make some operation easier (i.e., search for an identifier in a directory).

The *address* is a *name*, but the *name* is not necessarily *address*. The *address* is bound either to name(s) or object(s) in order to *locate* it(/them). Therefore, the *address* is always a pointer in *topology* (e.g., position of the node in the graph, grid coordinates). The *name* is merely a label without any context to location.

The **route** is the specific information needed to forward a piece of information to its specified *address*. Routing action may require one or a series of steps in order to forward information to reach a destination. There should be mechanism mapping *address* into an appropriate *route*.

Address is **location dependent** if it encodes (even the part of) *topology* information (i.e., *address* string depends on where the address is present in the topology). *Address*, which is **route dependent**, encodes (even the part of) *route* information. Because there may be more than one *route* to a given location, we want *addresses* to be *location dependent* but *route independent*.

3.2 Analogies

This subchapter summarizes system analogies also employing naming and addressing concepts.

3.2.1 Naming and Addressing in Telephony

The name in PSTN⁵² usually means the name of the customer (i.e. object) that is reachable via dialing a particular number, which could be considered as an address. Addressing in telephony service evolved from flat to (at least partially) hierarchical address *namespace*. Binding is administrative act conducted by telephone service provider that assigns a free number from its pool to the customer. Directories provide a mapping between names and addresses like a telephone book or white/yellow pages. To reach the object of a particular name in telephony means to dial its address (i.e. number) on phone board and wait until a connection is established. However, please notice that phone directory name is *address* as well, because for instance people with surname “Veselý” are grouped together (near each other).

Initially telephone users were interconnected using the single exchange office in a given area. The phone number corresponded with the slot/line number, which allowed only local calls. When the number of telephone users exceeded the line count on the exchange’s switchboard, multiple exchange offices interconnected with trunks were deployed. The original number was prefixed with unique exchange office code to differentiate between users with the same numbers but on different exchanges. Distance calls are allowed by introducing additional area or even country codes to the number.

Address	Name	Time Span	Description
6	Karel Chyba	1960 – 1972	Telephone link established to address Puškinova 634, Jeseník.
064 006	Karel Chyba	1972 – 1980	Until the year 1972, distance calls were available by using an operator that relied on a phone call to another exchange office.
0645 2448	Karel Chyba	1980 – 1986	Exchange office replaced to accommodate more users in a given area; this also included renumbering.
0645 2448	Jelena Veselá	1986 – 1990	The telephone changed the owner as an inheritance after deceased.
0645 202448	Jelena Veselá	1990 – 1992	Multiple exchange office installed in a given area, thus unique exchange code 20 was introduced.
00420 0645 202448	Jelena Veselá	1992 – 1993 1997 – 2001	International SS7 prefix introduced. Formerly it was 0042 for Czechoslovakia and currently it is 00420 for the Czech Republic.
00420 645 202448	Jelena Veselá	2001 – 2003	Removal of leading zero in the area code.
00420 584 413570	Jelena Veselá	2003 – onwards	Exchange office reinstallation together with the revision of the national numbering plan.

Tab. 5: Example of relationship between address and name in PSTN

⁵² Public Switched Telephony Network (PSTN). For more, see <http://en.wikipedia.org/wiki/PSTN>

Let us inspect this nature of phone names and addresses on a real-life example illustrated in Tab. 5. We can observe changes related to one particular telephone and line that was bought in the year 1960 and successfully operates to nowadays.

Telephone number as the address in telephone world is *location dependent*. Users share the same prefix (e.g., exchange/area/country code) that might be perceived as geographically nearby (violet, red and orange digits in Tab. 5). However, there are exceptions to this syntax like toll-free numbers (800) or emergency services (112, 150, 155 and 158) that should be considered location independent.

3.2.2 Naming and Addressing in Postal Service

The post is the nice example of the best-effort message delivery service. Envelope conventionally contains both, the name and the address of the receiver. The whole address is *location dependent* where each line specifies geographical location more and more precisely – country, ZIP code, city, district, street name, house number, and apartment/office. Once the letter is dropped into the post box, it is being routed and forwarded closer to the object (i.e., recipient's name) according to "envelope's PCI" (i.e., recipient's address). The post office aggregates routes based upon topology.

Unambiguity of the name in telephone or postal service depends on the scope in which we are inspecting its sui generis. For example birth name and surname may be enough for a small town but does not suffice in whole country scope. However, one person could have multiple names with the same address, for instance, senders might use the company name or its owner name to reach the same recipient. Also, multiple persons may have the same name and live on the same address.

Binding between name and address is fixed and predetermined by the location of the receiver. The mapping between name and receiver is unnecessary because one cannot exist without another.

3.2.3 Naming and Addressing in Operating Systems

Saltzer's paper [45] tries to provide the comprehensive theory of addressing and naming and applies it to programs and operating systems. The paper does not address computer networks directly, but many of the aspects are similar and applicable. By the term, the object could understand any data or computer program (or its parts).

Basic usage of names and addresses is a simple variable in any programming language. Variable's name is unique in particular scope (e.g., FOR cycle, procedure/function/method, the whole program), and it has some lifetime (e.g., until the end of the program, until garbage collector disposes of it). There are at least two variable's addresses: a) fixed logical address that references object; b) changeable physical address that is object's absolute placement in memory (allocated by the program or garbage collector). The value of variable could be accessed by its name or by its address (i.e., pointer).

The goal of naming is to allow **object sharing**, i.e. a) one object is a component of more than one other object; b) object may be used by two or more different, parallel activities at the same time. There are more objectives that we want to be accomplished by naming system:

- *Modular sharing* – A given object can be used without any prior knowledge of the names of objects that this object uses. Lack of modular sharing leads to name conflicts where we bind the same name to multiple different objects. This happens when we put together two different independent programs in a system, i.e., both of them operate under the same *namespace*;
- *Multiple-mappings* – One object could have multiple addresses where the reference to any of those addresses yields the same object. The good naming system should also deal correctly with unstable bindings (i.e. those that are changing during the time);
- *User-dependent bindings* – Different object's users should be able to access their components privately (e.g., the association between arguments to a function and its parameters). Thus, the same object may have multiple independent names, where each one is bound with different user-dependent accessibility. Nevertheless, those user-dependent bindings should not affect or conflict among each other.

A single object can have multiple names of different kinds, for instance, human-readable vs. computer-suitable name, local vs. global name, synonyms. Establishing of scope (context) for the name to address resolution can occur: 1) during compilation of program; 2) just before program first executes; 3) just before each execution; 4) during the execution.

The naming of files and programs in operating systems adopted a hierarchical approach using *pathname* that consists of the root directory, subdirectories, and local name. If we move a file from one folder to another, then pathname changes but the local name remains the same.

3.3 Theory

Employing knowledge from ISO/IEC 7498-3 [48], Saltzer's RFC 1498 [49] and Chapters 5 and 8 of Day's book [3], we will try to postulate some synthesis of the naming and addressing theory.

The *object address* is a *name* of the *object* to which it is *bound*. The *object* cannot be *located* without *identification*, nor can the *object* be *identified* without *localization*. Therefore, no reason exists, why to distinguish term *name* from *address* because *identifying* and *locating* the *object* are relations yielding same results. Hence, this means that *object name* and *object address* are same because they do not *identify* distinct *objects*. E.g., if "OBJ" is the *name*, then it is also its *address*, which help us to *identify/localize* an *object* in the scope of other objects. The previous statement is the final resolution of name-address dichotomy.

Three objects should be named in computer networks:

- 1) **services/applications** – Services are functions that are being used, e.g. service is Internet browsing. The application is using services, e.g. Internet browser. Difference between service, application and user are in this sense non-essential, and we are going to use them indistinguishably within this subchapter;
- 2) **nodes** – Nodes are (even virtual) computers that run services. Some nodes are hosts (service consumers) while other nodes provide auxiliary functions to run services (e.g., routing and forwarding by routers). When taking into account virtualization technique where one node could host multiple virtual nodes, more accurate term for node would be (N)-entity;
- 3) **network attachment points** a.k.a. PoAs – PoAs are (Internet-connected) interfaces/ports (i.e., (N-1)-entities) of a given node;

The natural way, how to relate to the preceding *objects*, is to use terms *application/service name*, *node address*, *network attachment point address*, even though that we could use *application address* or *network attachment point name* in compliance with this theory.

Following three *bindings* exist between *objects* above:

- 1) **directory** – Directory is *service* to *node* mapping used to find service's location (i.e., communication endpoint);
- 2) **routes** – Route is a sequence of *node addresses* calculated by the routing algorithm; route interconnects a given pair of source and destination *nodes*;
- 3) **paths** – Path⁵³ is a *node* to PoA mapping of the nearest neighbor (i.e., next-hop); path interconnects PoAs of adjacent *nodes*.

⁵³ Term "path" here differs from path known from graph theory.

Naming and addressing are free to use any form of identifier that seems helpful. It could be a binary or printable character string. The *namespace* and *address space* could be flat or hierarchical; the same *object* can even use different identifiers a.k.a. *aliases*, where some of them may be flat and others hierarchical.

Naming requirements (for more about them in frame of general networking, please see [44]) can be described in terms of *bindings* and *binding* changes among *objects* mentioned above:

- A given *service* may run on one or more *nodes*. Any *service* may need to move from one *node* to another without losing its identity;
- A given *node* may be connected to one or more PoAs. Any *node* may need to move from one PoA to another without losing its identity;
- One or more *paths* may connect a given pair of PoAs. Any of those *paths* may need to change without affecting the identity of the PoAs.

Each requirement contains some identity preservation, which is guaranteed when the *name* does not change during the moves – *object name* must be invariant when referring to some property of particular scope. This can be accomplished by maintaining a list of *bindings* between *services*, *nodes*, and PoAs. Basically, we name proper objects and then keep track of bindings between them.

To wit, *service/user names* do not change with location, *node names* do not change as PoA endpoints, and PoAs do not change as particular path endpoints. However, following rules do not mean that *names* should be assigned to a given *object* only once, and they cannot change after that. Essentially, *names* could be changed but this act must comply with previous requirements. Also, the *identity* of an *object* exists regardless of whether we can express it with some *name*.

If we want to send a packet to a given *service*, then following actions are done:

- 1) Find nodes on which the requested *service* operates. The task is *service name* resolution, which consists of *directory* search in order to discover a proper *binding* between *service* and *node(s)*;
- 2) Find routes between source and destination *nodes* and pick the next-hop *node*, where the packet should be forwarded. This process is a.k.a. **routing**, where the initial result is *route* as the sequence of *node names*, and following result is next-hop *node name*;
- 3) Find PoAs of the next-hop *node* en route, i.e., perform *node name location* to reach *node(s)* found in the previous step;
- 4) Find paths between the current and the next-hop *node*'s PoAs, i.e. discover the *binding* between the same PoAs pair and the *path*. This action is done by identifying a set of *paths* which leads among PoAs acquired in the previous step.

Each of previous steps might return either single or multiple alternatives. In the case of multiple returned *objects*, a choice must be made which of them to use. While these choices are distinct, they

might interact – e.g., we may swap communication to a different *node* running the same *service* according to the *path* aptness.

We can easily satisfy basic *object*'s properties using this theory – what it is, where it is and which way it is. To wit, when speaking of network *applications*, the *service name* provides an answer to *what*, node and PoA names provide answer to *where*, routes and paths provide answer to *which way*. The difference between *node address* and PoA *address* allows us to create a logical over the physical *address space* relation. A network addressing system must support at least one level of indirection.

Resulting model of this theory is illustrated in Fig. 13. This picture depicts a simple network with two levels of indirection; Internet use-case (as the network of networks) would require one more layer. Let us briefly inspect emerging properties of this model:

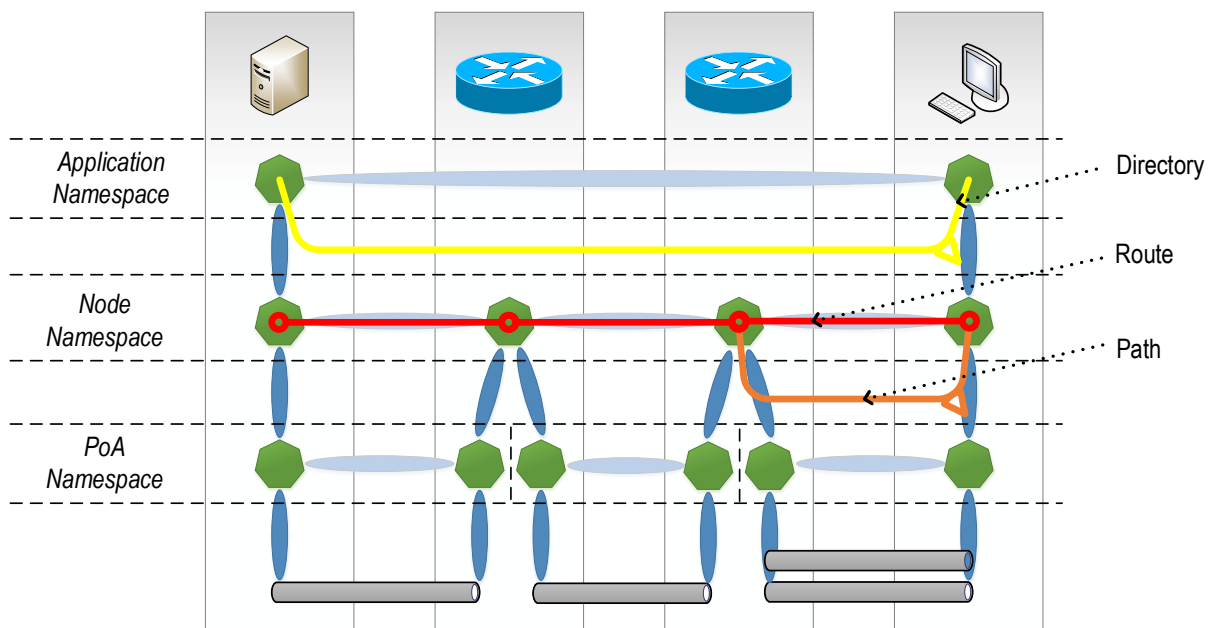


Fig. 13: Theoretical naming and addressing model for computer networks

- *Directory* and *path* mappings are similar in a way that both of them track the *binding* of *objects* one hop away.
- Two *nodes* could be interconnected via multiple distinct *routes* (containing different interim nodes).
- Two adjacent *nodes* could be interconnected via multiple distinct *paths* (separate physical connections).
- The *route* could be viewed as a concatenation of *paths* in a relaxed context.

The application name should be *location independent*. Node address should be *location dependent* (the logical address). PoA names are *route dependent* (the physical address). PoA *address* should be *unambiguous* only within a particular scope, and PoA *addresses* need not belong to the same *namespace* (e.g., Ethernet and FDDI addresses are from different *namespaces*).

3.4 Praxis

Despite the fact that Saltzer’s and Schoch’s papers are more than 30 years old and extensively cited, very few have been done to integrate their ideas into computer networking praxis. To wit, at least two following fundamental requirements exist for a correct addressing and naming system: 1) recognition of objects – applications, nodes, and PoAs; 2) distinguishing changeable bindings – application to node, node to route, node to PoA, and PoA to path.

Unfortunately, IPv4 does not follow those two requirements at all! Current Internet architecture contains only PoAs and routes; it completely misses application and node names. IPv4 address ought to identify a node, but it retains semantic of interface address. Unfortunately, this makes multihoming impracticable because IPv4 address labels only node’s PoA not a node itself. What is worst, IP address names the same thing as MAC address. Routes are then falsely bound to an IP address. Instead of the general directory, the Internet is stuck with well-known port numbers (SSH is on 22, Telnet on 23, SMTP on 25, HTTP on 80 and so on) and they are no more than a suffix to the network address. Basically, a node layer is missing.

On Fig. 14 current broken model is depicted:

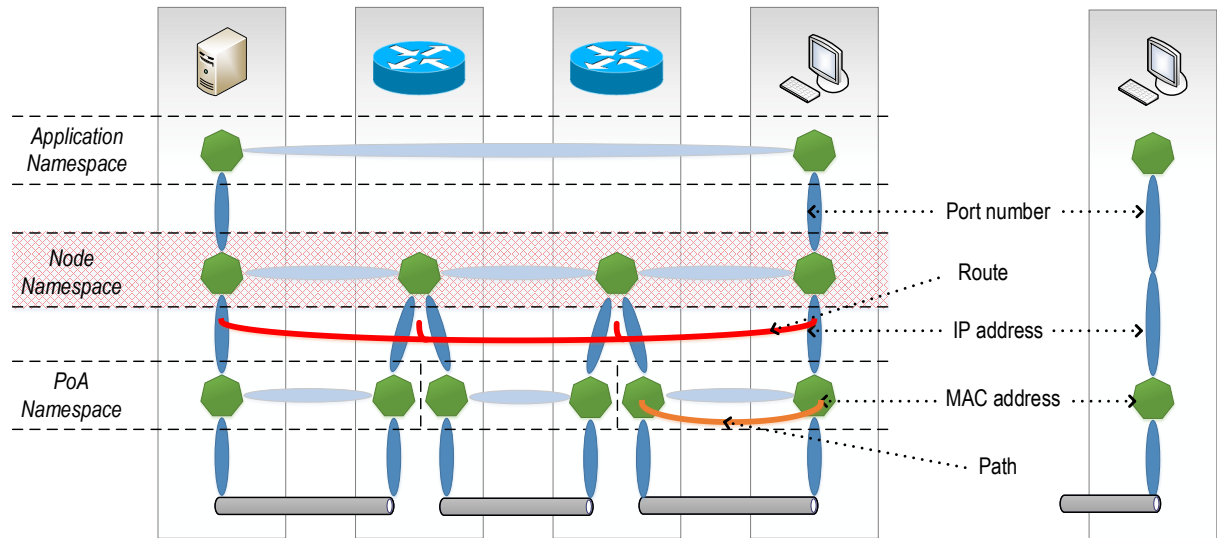


Fig. 14: Broken Internet naming and addressing model

The previous figure illustrates that major flaw exists in current TCP/IP naming architecture. Mostly because of this poor design, Internet suffers from issues described in Subchapter 2.2. The following attempts tried to provide some redemption, and each one of them is touching some part of the overall problem. Interesting is that working, accepted and implemented alternative with complete naming (comparing to TCP/IP) architecture for computer networking already existed – OSI-RM (namely its part [48]). However, OSI-RM’s experience was not used as a design guide for the development of IPv6, DNS or URIs.

3.4.1 Internet Protocol Version 6

As RFC 4292 [50] clearly states, IPv6 addresses are identifiers of (a set of) interfaces, not nodes. Hence, the same problem leads to the same troubles as in IPv4. An integral part of one of the Saltzer's naming requirements is missing in IPv6. However, let us not condemn IPv6 without proper inspection of what is named and what is addressed by IPv6.

IPv6 gets rid of broadcast (one-to-all) and establishes following three kinds of communication which are closely coupled with the particular address type:

- *Unicast* (one-to-one) – IPv6 address identifies a single interface. A packet is delivered to a single host only;
- *Anycast* (one-to-nearest) – IPv6 anycast address identifies a set of interfaces (usually on different nodes). A packet is delivered to one of the interfaces from a set. There is no syntax difference between anycast and unicast addresses;
- *Multicast* (one/many-to-many) – IPv6 multicast address identifies a set of interfaces. A packet is delivered to all interfaces from a set.

IPv6 allows an interface to have more than one IPv6 address, and any interface has at least three addresses (one link-local, two multicast addresses). There is a long list of specific IPv6 types of addresses briefly summarized in Tab. 6.

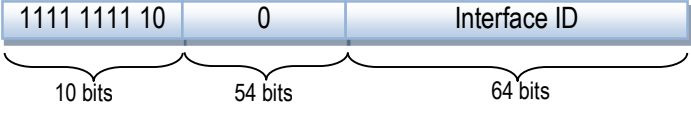
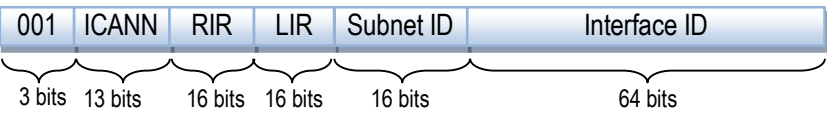
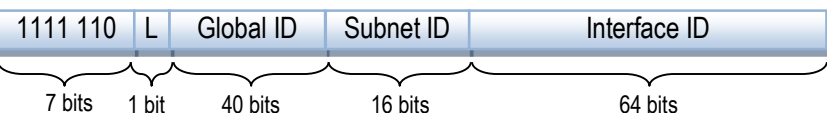
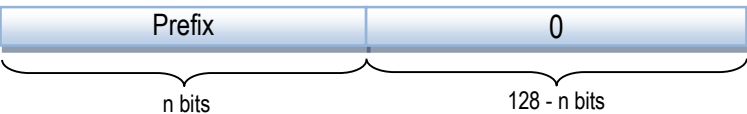
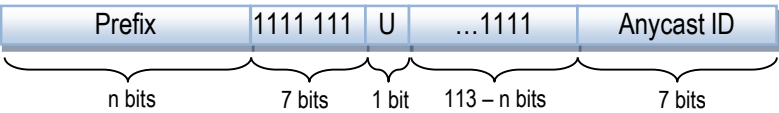

Michael O'Dell's GSE addressing architecture [51] divided IPv6 address into two parts – one serves as a routing locator and another as an identifier. Let us focus on the unicast address, because only unicast prefixes are present in DFZ routing table. Routing part consist of *ICANN*, *RIR*, *LIR* and *Subnet ID* fields that allow to create a hierarchy. However, address with this property is aggregatable only when it is PA address. Identifier part (called *Interface ID* in Tab. 6) could be: 1) set statically; 2) derived from computer's MAC address⁵⁴; 3) generated cryptographically⁵⁵; 4) changed randomly⁵⁶. Unfortunately, neither way can guarantee global identifier unambiguity, thus routing part is always needed as a fail-safe against identifiers collision.

We observed some trends in aggregated and deaggregated prefix counts in Subchapter 2.3. The hope for DFZ is to decrease the amount of deaggregation. The IPv6 address is assigned to the interface (not a node), and when IPv6 tries to embed the unambiguous identifier in the address, it fails. Hence, we are back at square one with IPv6, when dealing with problems such as multihoming or mobility mentioned in Subchapter 2.2.

⁵⁴ This kind of identifier is called EUI-64. See <http://standards.ieee.org/regauth/oui/tutorials/EUI64.html>

⁵⁵ Identifier is generated using asymmetrical cryptography, thus providing how to verify host identity. For more, see RFC 3972.

⁵⁶ To ensure anonymity during IPv6 communication, hosts are able to generate own IPv6 addresses and change them as they want. For more, see RFC 3041.

Type	Syntax / Format / Description
Unspecified	::0/128 This address cannot be assigned to an interface. It is used only during initial enrollment of the interface to the network.
Loopback	::1/128 The node uses this address to send a packet to itself. Equivalent of IPv4 127.0.0.0/8 addresses.
Link-Local	fe80::/10  <p>Every interface has at least one link-local address that allows communication with all adjacent (Hop Count = 1) devices.</p>
Global Unicast	2000::/3  <p>Currently only prefix 2000::/3 is assigned for global unicast communication, where usual address starts with “2001:”. However, some transition mechanisms introduced additional prefixes (2002: for 6to4).</p>
Unique Local Unicast	fc::/7  <p>Equivalent of private IPv4 addresses⁵⁷ intended for limited, organizational scope communication.</p>
Anycast	  <p>As mentioned before, anycast addresses are part of unicast address space. However, conventions reserve part of this subspace for potential anycast.</p>
Multicast	ff00::/10  <p>The multicast address identifies a group of interfaces (Group ID). This group identifier could be created in multiple ways, thus introducing a variety of multicast addresses with different syntax – unicast-prefix-based, source-specific-multicast, interfaceId-based or embedded-RP addresses.</p>

Tab. 6: IPv6 address types

⁵⁷ Private IPv4 addresses are reserved for private usage by any organization. The list available at RFC 1918.

3.4.2 Domain Name System

Domain Name System (DNS) was invented as a descendant of the former `hosts.txt` file⁵⁸, which distributed a list of IPv4 addresses of devices connected to the Internet. DNS is distributed mapping system of records providing resolution of domain names to IP addresses (i.e., A- and AAAA-record). Apart from that, DNS supports, reverse resolution (i.e., PTR-record for IP address to name), address of mail servers (i.e., MX-record), domain name synonyms (i.e., CNAME), geolocation information (i.e., LOC), service locators (i.e., SRV), storage place for variety of keys, signatures and certificates and many more.⁵⁹ Fully qualified domain names (FQDN) create DNS's *namespace*. Each FQDN consists of multiple hierarchical parts delimited by dot character – starting from the right the first one is top-level domain (abbreviated TLD as top-level domain, e.g., cz, com, net, sk), followed by first-level domain (e.g., company name or web service name), followed optionally by subdomain (e.g., organization's department) and finally followed by a device's hostname. Example and FQDN syntax are depicted in the Fig. 15

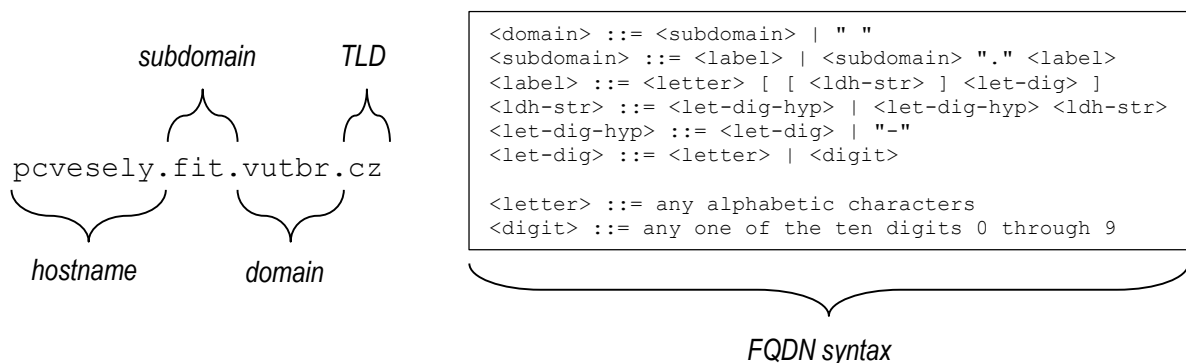


Fig. 15: Fully qualified domain name example and syntax

However, FQDN is not an equivalent to a Saltzer's node name. DNS does not provide a binding between node and PoA because current TCP/IP architecture of Internet lacks it. For instance, the device retrieves multiple FQDN-to-IP mappings for a single query and chooses between them in a round-robin fashion. Each returned A-record represents original PoA, but PoAs might be not only from the same but also from different devices. Thus, the same FQDN cannot distinguish between the various devices as an unambiguous node name. Hence, FQDN is nothing more than a (both direct and indirect) alias for IP address from a different *namespace* (readable to humans).

⁵⁸ For more about hosts file, please see [https://en.wikipedia.org/wiki/Hosts_\(file\)](https://en.wikipedia.org/wiki/Hosts_(file)).

⁵⁹ For details about DNS resource records, see RFC 1035 and related RFCs.

3.4.3 Uniform Resource Identifier

Uniform Resource Locators (URL) and Names (URN) later replaced by Uniform Resource Identifiers (URI) are just a next step of naming evolution started by DNS and inception of World Wide Web (WWW) service. URI is a string that identifies the abstract or physical resource. URI allows uniform (i.e., uniform semantic interpretation) identification (i.e., to distinguish between other objects within the same scope) of resources via an extensible set of naming schemes. URIs also have a hierarchical structure that consists of multiple components – mandatory *scheme*, optional *authority*, mandatory *path*, optional *query* and optional *fragments* delimited by colon, slash, exclamation and hash characters. Fig. 16 depicts URI's structure and also some examples.

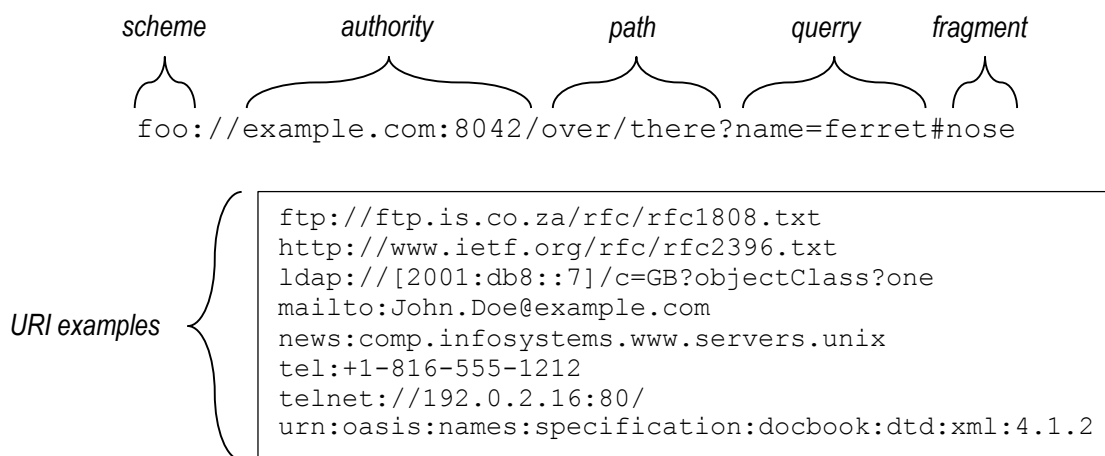


Fig. 16: Uniform Resource Identifier structure and examples

URI's *scheme* part provides application type (e.g., telnet, FTP). However, another part is URI *authority* that is either FQDN or IP address followed optionally by port number. Because of that, URI cannot be treated as the *application name* but sort of the path to the application. Regarding that, URI changes whenever *application* moves to other *node*. Hence, all previously mentioned problems of IP addresses and DNS influence URI proper usage, so that URI itself does not satisfy Saltzer's naming requirements.

3.5 Possible Solution

This subchapter succinctly sums up theoretical properties of any solution based on Subchapter 2.2 analysis and RFC 6227 [17]. Moreover, it describes and compares features of some existing candidates.

3.5.1 Ideal Solution Properties

Following section is based on IETF's observation of current Internet problems and desired qualities. Unfortunately, it does not necessarily comply with theory that we have outlined and investigated in Subchapters 3.1, 3.2 and 3.3. However, following description provides at least some guideline how to compare feasibility of candidates.

One of the major goals for any upcoming change of the Internet architecture is to make the routing system scalable with respect to a number of prefixes, users and interconnections between autonomous systems.

As stated above overloading of IP address semantics causes collisions and limited flexibility. Hence, it is expected that a solution would decouple identifier namespace from location address space. Nevertheless, there are two approaches how separation should be performed: a) by splitting hosts, identifiers, and locators; b) by removing end-site prefixes from globally routable prefixes. The solution should contain the fix and should be compatible with either case. Ideally identifiers should be allocated at the birth of object, they never change, nor are they re-used (but take into account that one would need infinitely long address to truly achieve this goal). Hence, identifiers must be location-independent. Locators should point to device's position in the network, and they should change whenever the graph changes, thus locators must be location dependent.

The more scalable solution for multihoming is strongly desired to allow organizations multihome without adding pressure to DFZ routing tables.

As for mobility more efficient approach is wanted that allows mobile entity topological changes at a high rate. Hypothetically ideal solution should decouple mobility completely from routing.

TE is a necessity for a network operation of any organization. However, solution for inbound traffic engineering should pose no burden to the scaling of the routing system.

Renumbering is an inconvenience for either small or large scale networks. Even with the existence of working methodologies like RFC 4192 [52] how to renumber without a Flag Day it is still difficult to make this process cheap and smooth for any organization. Therefore, it is required that organizations could renumber their networks easily with as less disruption as possible.

Previous features refer to existing and above thoroughly described issues. Nevertheless, there are two more properties, which any solution should incorporate. The routing system is secured through additional protocol-specific mechanisms (i.e. mutual authentication of routing updates with the help of HMACs) that were introduced later during target routing protocol lifecycle. Hence, the solution must provide the same level of routing security, or better it must be secure by design. Also, any solution must

be deployable from technical and practical perspective – it must allow incremental deployment and provide necessary backward compatibility with currently deployed services.

Any possible solution should somehow address all above-stated problems. RRG even prioritizes them by a degree how mandatory the fix supposes to be a part of the new architecture. Ladder of obligation is as follow: REQUIRED (which means that solution must support this goal) > STRONGLY DESIRED (which means that solution should support this goal unless there is a good reason not to do so) > DESIRED (which means that solution should support this goal). Tab. 7 provides summarization of community consensus:

Abbreviation	Design Goal	Priority
RS	Routing Scalability	STRONGLY DESIRED
DIL	Decoupling Identification and Localization	DESIRED
MH	Multihoming	STRONGLY DESIRED
Mob	Mobility	DESIRED
TE	Traffic Engineering	STRONGLY DESIRED
Ren	Renumbering	STRONGLY DESIRED
Sec	Routing Security	REQUIRED
Dep	Deployability	REQUIRED

Tab. 7: Design goal importance for a new routing architecture

3.5.2 Existing Proposals

RFC 6115 [53] clearly states that: a) RRG has rough consensus on separating identity and location of devices but does not have consensus how to do it properly; b) RRG has consensus that multihoming and traffic engineering issues need to be solved in a scalable manner.

Theoretically, there are three ways how to decouple identity and locality:

- *Map-and-encap network-based architecture* – It evolves from Robert Hinden’s ENCAPS protocol [54]. When a source sends the packet towards destination outside of source network, the packet must traverse through border router between two address spaces (locator space and identifier space). Here at first border router performs mapping of an identifier to appropriate locator (“map” phase). Then the packet is encapsulated using returned locator address (“encap” phase). Hence, map-and-encap principle wraps a new header (called *outer header*) using locator addresses around the original header (called *inner header*) with identifier addresses. When encapsulated packet reaches the destination network, the border router strips off the outer header and sends the original packet towards the receiver. Map-and-encap usually does not require changes to hosts or to the core routing infrastructure (that is DFZ). Unfortunately, with additional overlay encapsulation comes size overhead.
- *Rewriting hybrid network-based architecture* – Originally this principle comes from papers written by Robert Smart and David Clark 8+8 [55] and later by Mike O’Dell GSE [51]. It utilizes

IPv6 field so that upper part of IPv6 address is locator and the lower part stores an identifier. If a source sends packet outside its domain, border router takes addresses containing only identifiers and fills upper bits with appropriate locators. Then locators are removed from addresses upon reception by destination border router. Rewriting schemes may differ whether they perform either destination or both destination and source addresses rewrites;

- *Host-based architecture* – Decisions in this architecture are purely in the hands of hosts. Thus, hosts prepare and fill all relevant PCI fields (including locators and identifiers) as the packet is being dispatched by the operating system. Interim devices like routers are usually transparent to this approach.

According to [56], possible solutions could be categorized into two classes that are not opposites. Over the years following terms were established to describe them:

- **Core-Edge Separation (CES)** – A subset address space (*edge*) corresponding to end site addresses is separated from the transit DFZ (*core*). This “edge” address space is then handled differently for routing. Subsequently DFZ routing table increases its size only a new ISP transit network instead of a new edge network. Some mapping system is needed to glue core and edge address spaces. CES is depicted schematically in Fig. 17 where it shows communication between *PC-A* and *PC-B* using (green) identifiers and (red) locators;

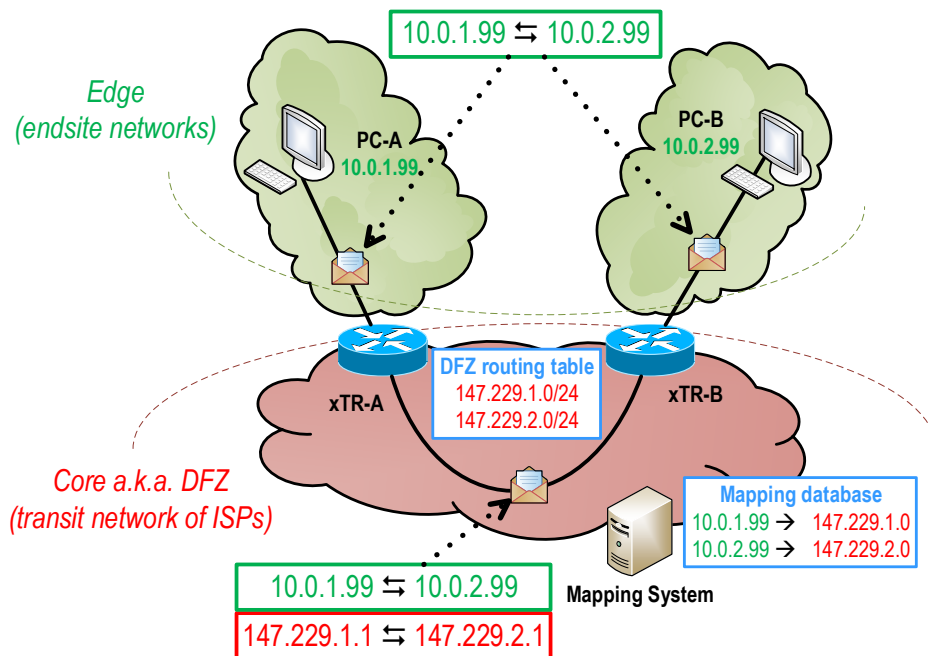


Fig. 17: Core-Edge Separation solution

- **Core-Edge Elimination (CEE)** – The goal of CEE is to eliminate all PI and de-aggregated PA prefixes from the core. Hosts then use either PA addresses provided by ISPs or usually something different (not in IP address namespace) as an identifier. Some changes in host network behavior are necessary to deploy CEE. Illustrated in Fig. 18.

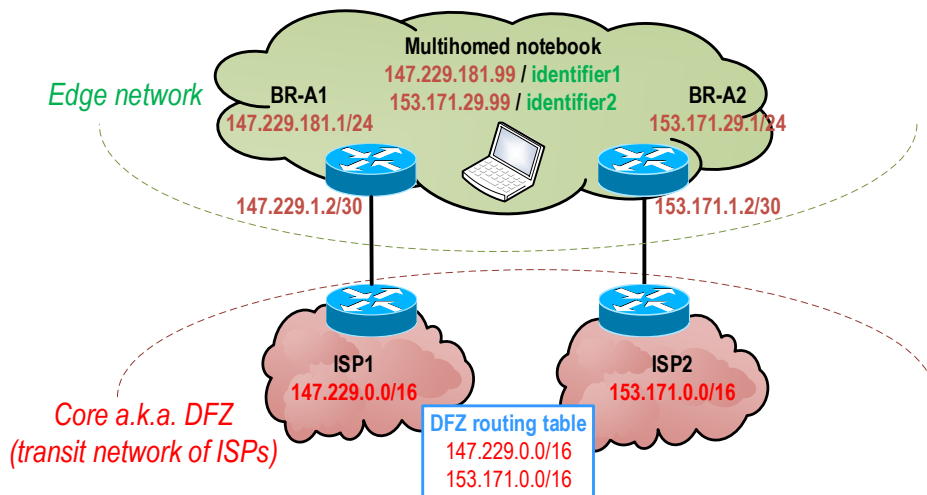


Fig. 18: Core-Edge Elimination solution

Down below is the short list of solution candidates. It is outside the scope of this thesis to describe them in depth. Hence, the astute reader is advised to follow bibliography links. Neither is this dissertation able to cover all candidates.

Locator/ID Separation Protocol

LISP focuses on the separation of locators and identifiers into two distinct address spaces using mapping and encapsulation on routers residing on the borders between those two spaces. Only locators are present in DFZ, thus are a possible subject of topological aggregation. With the separation of identifiers comes the ability to renumber cost effectively. LISP contains by design traffic engineering techniques so that more-specific prefixes could be removed from the global routing table.

When using LISP, there is no need to change anything on hosts or DFZ routers. LISP has well-defined deployment plan and interoperability with existing Internet architecture. LISP is beneficial to adopter since the first day. Moreover, implementations already exist and are undergoing testing in SOHO and also enterprise environments.

LISP utilizes robust mapping system based on a pull model, where queries are data driven. However, it may introduce an additional delay or even packet losses when the identifier-to-locator mapping is being discovered. Also, reachability and liveness of locators are not yet sufficiently resolved issues.

Chapter 4 covers LISP in more detail – operational principles, syntax, and semantics of control messages and hypothesis how to improve its functionality as one of the main contributions of this dissertation. LISP employs map-and-encap principle and it is CES solution.

Host Identifier Protocol

Host Identifier Protocol (HIP) is host-based approach how to perform locator/id split. Network layer employs IP address as locator, transport and application layer uses the identifier in the form of the cryptographic private-public key pair. Each host handles this kind of pair generation. *Host Identity Tag*

(HITs) – 128 bit long hashed public part of identifier pair – is used for communication and stored in the extension header. HIP makes use of DNS or distributed hash table (DHT) to obtain the identifier.

Among advantages of HIP is that it allows mobility and multihoming across different address families. HIP offers end-to-end encryption via IPsec. Most notably, it moves away from binding application to IP addresses.

When traversing NAT, HIP needs rendezvous server or sponge on Teredo. However, both of those approaches introduce unnecessary triangle routing between parties. Critique of HIP points out that HITs are without any inner structure, thus creating a flat *namespace*.

HIP is host-based CEE solution. More about HIP in RFC 4423 [57] and RFC 5201 [58].

Level 3 Multihoming Shim Protocol for IPv6

Level 3 Multihoming Shim Protocol for IPv6 (Shim6) splits locator/id in a manner that IPv6 PCI field address contains locator and extension header contains *Upper Layer ID* (ULID). It is the host-based solution with network layer approach working per host pairs rather than per transport layer session. ULID is used by upper layer protocol (i.e. TCP or UDP). When current locators become unavailable, Shim6 looks up for new locators and rewrites IPv6 addresses, thus providing session survivability.

DNS queries provide a possibly incomplete set of locators to hosts. It employs initial 4-way handshake during which locator sets are also exchanged. Keepalive mechanism is used to track locator's reachability.

Shim6 allows host-multihoming not site-multihoming also traffic engineering is not a part of Shim6 standard because TE does not concern hosts.

Shim6 is host-based CEE solution. More about SHIM6 in RFC 5533 [59].

Routing Architecture for the Next Generation Internet

Routing Architecture for the Next Generation Internet (RANGI) append one new layer between network and transport layer just as HIP. Hence, flows and connection are bound to host identifier instead of IP address that now serves as a locator. Unlike to HIP, RANGI host identifiers are hierarchical in organized structure. RANGI appends identifiers as special IPv6 destination options header, and locators are embedded as special IPv4 address into IPv6 PCI fields. RANGI utilizes: a) DNS for the translation of FQDN onto host identifiers; b) hierarchical DHT for host identifier to locator mapping.

Routing scalability is accomplished by decoupling locators and identifiers. Mobility and multihoming are supported because communication is now bound to an identifier, not a locator. Thus, sessions are not interrupted due to locator change or failure in redundant scenarios. Hosts might suggest TE while *Locator Domain Border Router* has the authoritative power to enforce TE. Deployment and compatibility with current Internet architecture leverage ISATAP⁶⁰ tunneling principle.

⁶⁰ Intra-Site Automatic Tunnel Addressing Protocol (ISATAP). For more, see RFC 5214.

The most severe disadvantage of RANGI is that change to TCP/IP stack is necessary for devices. Special proxy routers (called *Site Proxy* and *Transit Proxy*) are needed for communication with legacy hosts. Also, cryptography is seen as an issue for devices that are incapable of or do not want to support crypto algorithms.

RANGI employs host-based principle and is CEE solution. More about RANGI in [60] and IETF's Work in Progress papers [61] and [62].

Internet Vastly Improved Plumbing

Internet Vastly Improved Plumbing (abbreviated Ivip, pronounced [arvip]) is another locator/identifier splitter. It works with map-and-encap principle same as LISP. However, Ivip uses global mapping system instead of hierarchical LISP pull model. Mapping changes are propagated to *full database query servers*, which could be run by ISPs and/or end-sites. These servers create distributed network of cross-linked multicast trees. To reduce the load, new mappings could be cached by *query servers with the cache*. The difference from LISP is that Ivip maps always only single locator to a given identifier and mappings are updated in real-time. Ivip employs direct IP-in-IP encapsulation unlike LISP's interim header between inner and outer IP.

One of doubtful consideration of Ivip is whether global mapping database could attain real-time synchronization. Also, Ivip is missing clear deployment plan that can work without Flag Day and huge investments in resources.

Ivip employs map-and-encap principle and is considered to be a CES solution. More about Ivip on Ivip website [63] and in papers [64], [65], [66], [67], [68] and [69].

Hierarchical IPv4 Framework

The Hierarchical IPv4 Framework (hIPv4) introduces an additional hierarchy of IPv4 address space by dividing it into *area locators* (ALOCs) and *endpoint locators* (ELOCs). ALOCs are globally unique; ELOCs are unambiguous only locally. ALOC and ELOC are inserted as PCI fields into new shim header that resides between network and transport headers.

Instead of tunneling, hIPv4 employs swapping of addresses inside IPv4 header with ALOCs and ELOCs in shim header (appended to IP as a new PCI option field). This swapping is performed by dedicated routers called *Locator Swap Router* (LSR) which resides in ISP's ALOC realm. LSR RIB contains only ALOCs and local ELOCs. When ISP migrates its network to an ALOC realm, only ELOCs are exchanged via routing updates with LSRs from other realms.

The hIPv4 utilizes DNS for distribution of ELOC to ALOC mappings. To support multihoming and TE, hIPv4 must be combined with transport protocols such as MPTCP⁶¹ and SCTP⁶². ALOC (so to say PoA) is returned upon DNS request for a given ELOC – more than one ALOC might be retrieved.

⁶¹ Multipath TCP (MPTCP). For more, see <http://tools.ietf.org/wg/mptcp/>

⁶² Stream Control Transmission Protocol (SCTP). For more, see RFC 4960.

As with all locator/id splitters, it is easy to renumber sites when changing ISPs because only different ALOC is mapped to the same ELOC.

The major disadvantage is that TCP/IP stack requires a change for devices communicating with the non-hIPv4 world. This means that benefits for hIPv4 adopters will be apparent only after the majority of devices migrate. Besides that, also change is needed for some application protocols that convey IP addresses in its SDUs. Another negative feature is that hIPv4 takes into account only IPv4 address space; there is no support for IPv6 addressing.

The hIPv4 employs rewrite principle, and it is neither CES nor CEE solution. More about hIPv4 in RFC 6306 [70].

Name Overlay Service for Scalable Internet Routing

Name Overlay Service for Scalable Internet Routing (NOL) adds to TCP/IP stack new functions that manage configuration, registration and authentication of host names together with management of transport channels using those names and mobility for data transport. NOL utilizes session layer between transport and application layers. It uses rewrite principle, which introduces a new device called *Name Transfer Relay* (NTR) that carries out translation between reserved PI addresses representing names and globally routable PA addresses. This separation prevents PI prefixes from entering DFZ, thus reducing DFZ routing table size. Legacy devices accessing NOL-devices use special NOL proxy or to assign some of the globally routable PA addresses to specific servers behind NOL.

There is no requirement to change TCP/IP stack and no need for a new mapping system. NOL make use of DNS by storing name as a new kind of DNS record. The name is similar to email address `hostname@example.com`. Entry in DNS exists for `@example.com`, which points to NTR's PA address. PI address of the `hostname` is known only to NTR. Enforced utilization of PI addresses avoids the need for any renumbering. The mobility of transport sessions is achieved by checkpointing sequence numbers, but it works only between NOL-enabled hosts. NTR deployment is unilateral just as NAT.

Despite the fact that TCP/IP stack is left intact, applications on host need to be re-implemented to support NOL. Rapid updates to DNS's name-to-NTR mapping are needed when considering functional NOL multihoming scenario between different NTRs.

NOL employs rewrite principle and is neither CES nor CEE solution. More about NOL in [71].

Global Locator, Local Locator, and Identifier Split

Global Locator, Local Locator, and Identifier Split (GLI-Split) decouples identifiers and locators in a undermentioned manner. It differentiates between *global locators* (GLs) used in DFZ and *local locators* (LLs) used in edge networks. Besides that, GLI-Split also presents *static identifiers* (IDs) to identify endpoints of communication. Locators and IDs are embedded into IPv6 addresses, thus allowing backward compatibility with the IPv6 world. The higher 64 bits of GLI-formed IPv6 address contain

locators; the lower 64 bits contains an identifier. It encodes two different *namespaces* (each one 64 bits or less) onto single IPv6 address.

Separation of core and edge routing helps to aggregate prefixes. As any other locator/id splitter, renumbering is not an issue for GLI-Split. Besides that, internal rearranging of local locators is not visible globally. In comparison with LISP, Ivip or NOL, communication with legacy Internet is without any proxies or stateful NATs. GLI-Split uses global (i.e. ID-to-GL) and local (i.e. ID-to-LL) mappings, where global mappings leverage DNS.

The major criticism of GLI-split is that host TCP/IP stack change is required to interpret appropriately GLI-Split address and possibly perform mapping lookups. However, no changes to the application are needed in opposite to other CEEs. GLI-Split uses rewriting instead of map-and-encap. Thus, no additional state is needed for devices, where rewriting occurs. Moreover, as with all proposals depending on DNS, there is always an issue with the updating speed of DNS.

GLI-Split is CEE solution that employs rewriting. More about GLI-Split in [72] and [73].

Tunneled Inter-Domain Routing

Tunneled Inter-Domain Routing (TIDR) is locator/id splitter that is employing dedicated tunnels at the borders of DFZ. It works as an improvement of BGP that defines new attributes used for a distribution of identifier-to-locator mapping.

Identifier prefixes are stored in a new control plane structure called *Tunnel Information Base* (TIB). When a packet to identifier prefix is being routed, first TIB is searched to perform tunneling followed by RIB lookup regarding the routing. All interim routers route packet until it reaches tunnel endpoint where it is decapsulated. TIDR improves BGP convergence time for the specific scenarios. It supports TE and limited multihoming by design and as such it spares depletion of ASN *namespace*.

Despite the fact, it reduces RIB, TIDR only offloads information from RIB to TIB. Moreover, it does not take into account FIB whatsoever. Hence, it does not help with the scaling problem to accommodate the increasing number of organizational networks. Also, TIDR benefits will not be apparent unless all DFZ routers migrate to TIDR.

TIDR is a CES solution. More about TIDR in IETF draft [74] and mailing list (namely [75] and [76]).

Identifier-Locator Network Protocol

Identifier-Locator Network Protocol (ILNP) decouples identity and locality inside IPv6 PCI field. First 64 bits are used as locator name that might change; remaining 64 bits are used as a node name. Applications bind only to identifiers, which remain constant during a lifetime of transport layer session. Multiple locators might be used by a node simultaneously. ILNP insists on the establishment of new DNS records to support node backward/forward resolution of locators/identifiers to FQDNs.

ILNP supports site and also node mobility and multihoming. No changes are needed to exist DFZ routers, and it has well-stated incremental deployment plan. As with other solutions splicing on DNS, ILNP hurdle is a silent expectation of near-zero time to live of some DNS records and their maintenance.

ILNP is CEE solution using rewriting principle. Development of ILNP is pursued further by IETF and its RRG. More about ILNP on project website [77], previous draft [78] and subsequent RFCs 6740-6748 [79], [80], [81], [82], [83], [84], [85], [86] and [87].

Evolution

Rather than a new architecture, Evolution is the best-practice proposal. Evolution employs the idea of applying *FIB Aggregation (FA)* with increasing scopes to evolve more scalable routing system. Unlike CES proposals, Evolution does not start with some predefined border between core and edge networks. Aggregation scopes start from the single router, and then to single network, ending with aggregation of neighbor networks.

Evolution is stepwise process consisting of following phases:

- 1) FA on a single router where FIB is algorithmically compressed without changing RIB. Software upgrade is needed for this;
- 2) Intentional configuration of provider edge routers, autonomous system boundary routers, and BGP route reflectors as next-hop-self default gateways for a given AS;
- 3) *Virtual Aggregation (VA)* in a single network where some routers in AS are marked as *Aggregation Point Routers (APR)*. APRs maintain full FIB table, others may suppress some of their FIB entries and deliberately route packets to APRs;
- 4) VA across neighbor networks that also applied VA so that in BGP updates path to egress router is available directly;
- 5) Reduction of RIB size by outsourcing control plane to external controllers, which perform eBGP peering (and provide necessary information) to forwarding DFZ routers;
- 6) Isolation of DFZ routers from routing churn for instance by handling certain prefix inaccessibility locally.

Evolution proposal is comparing to others easiest for deployment with immediate impact for adopters. Among concerns is that improperly accomplished Evolution may introduce routing loops or **reverse path forwarding (RPF) check**⁶³ failures. On the contrary to other proposals, Evolution does not address mobility.

⁶³ **Reverse Path Forwarding (RPF) check:** When using RPF check, packet incoming interface is checked whether it is the same one as outgoing interface towards a network of sender by a routing table. If it is true, then packet is forwarded, otherwise it is dropped. See <http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html>

Evolution is neither CES nor CEE solution. Development of Evolution is pursued further by IETF and its RRG. More about Evolution in [88], [89] and [90].

Name-Based Sockets

Name-Base Sockets (NBS) are a new alternative for socket-based communication. Unlike nowadays sockets (e.g. BSD sockets) that are bind to IP addresses, NBS are bind to domain names as their name suggests. As consequence, applications start to communicate using domain names as endpoint selectors where appropriate IP address re-/selection is left on TCP/IP stack itself.

NBS helps organizations to prefer PA address by making them more acceptable to use for multihoming and less avoided for renumbering. Thus, NBS decrease reliance on PI addresses.

A necessary prerequisite for NBS is their adoption by host's operating systems (OS) which is also its major disadvantage due to the usual inflexibility of OS vendors. Existing applications should be augmented, and new applications developed directly using updated socket API⁶⁴ to profit from NBS. However, there is an immediate benefit for NBS adopters. NBS deployment is incremental and does not pose any threat to legacy applications.

NBS is CEE solution that does not use neither map-and-encap nor rewrite principle. More about NBS in [91] and [92].

A Practical Transit-Mapping Service

A Practical Transit-Mapping Service (APT) is similar to LISP that it is CES solution using map-and-encap principle with additional UDP header. Tunnel routers for LISP are customer's edge devices, for APT they are provider's edge – the APT is more ISP-centric.

Instead of a globally available hierarchical mapping system, all APT-enabled AS has *default mappers* (DM) that periodically synchronize. Mapping information is then retrieved using local pull to default mapper. APT tries to handle packet loss by rerouting between DMs, which also maintains reachability status of RLOCs.

New BGP attributes carry EID-to-RLOC mappings between peering DM. However, mapping announcements must be cryptographically signed to be accepted by DM. This is to limit mapping corruption or spoofing in APT, but it is also one of the major disadvantages.

The development of APT is no longer active. More about APT in [93].

Internet Routing Overlay Network with Routing and Addressing in Networks with Global Enterprise Recursion

The Internet Routing Overlay Network with Routing and Addressing in Networks with Global Enterprise Recursion (altogether IRON-RANGER) uses IRON routers that interconnect recursively-nested RANGER networks. IRON-RANGER utilizes own tunneling and path MTU discovery

⁶⁴ Application programming interface (API). For more, see <http://en.wikipedia.org/wiki/API>.

(operation of MTU discovery along the path is often abbreviated as PMTUD) management protocol called Subnetwork Encapsulation and Adaptation Layer (SEAL) [94] for separating identity of the node from its locality. IRON-RANGER is architecturally derived from ISATAP. From the IRON-RANGER point of view, DFZ is understood as one non-broadcast multi-access (NBMA) network. IRON-RANGER utilizes two approaches: a) proactive routing protocol distributes highly aggregated virtual prefixes (VP); b) data-driven protocol distributes more specifics into IRON router's FIBs.

A major criticism of IRON-RANGER is that protocol SEAL is rigidly using *ICMP Packet Too Big* and *ICMP Fragmentation Needed* messages to enforce sizes typically below 1500 B, thus preventing any jumbo grams. It does not provide true location independent identity. These together with mobility as disadvantages are left to other disjunctive proposals that can cooperate with IRON-RANGER (e.g. HIP, RANGI).

IRON-RANGER is CES solution with map-and-encap principle. More about IRON-RANGER in RFC 5720 [95].

Tunneling Route Reduction Protocol

Tunneling Route Reduction Protocol (TRRP) interconnects tunnel routers (xTR) through GRE⁶⁵ tunnels. Other BGP-peering routers point their default routes towards xTRs which perform DNS lookup to find endpoint tunnel destinations.

TRRP does not need new DNS records, instead of that it redefines the meaning of TXT record to carry information regarding the feasibility of router (to prefer or to avoid it) and applicable GRE mode (direct, GRE over IPv4 or GRE over IPv6). Despite the fact that TRRP cleverly reuses existing technologies, it has some disadvantages. TRRP does not take multicasting into account, and its proposal provides no mentions about multihoming.

The development of TRRP is no longer active. TRRP employs map-and-encap principle and is CES solution. More about TRRP in [96].

Six/One Router

Six/One is yet another CES solution utilizing rewriting principle on the border so-called *Six/One Routers*. It separates edge as *local addresses* and core as *remote addresses*. Six/One takes advantage of the special IPv6 extension header.

It helps with routing scaling, renumbering concern and multi-homing but does not address any mobility issue. Besides that, Six/One does not have any interim device that can mediate communication between Six/One and non-Six/One host. Also as another disadvantage, it supports only IPv6 address family. Mapping system sponges on DNS where it assumes the definition of new resource records, but no detail specification is provided.

The development of Six/One is no longer active. More about Six/One in [97].

⁶⁵ Generic Routing Encapsulation protocol (GRE). For more, see RFC 2784.

Recursive Internet Architecture

All previous proposals boldly consider themselves as another or even new Internet architecture. As it is apparent from this chapter introduction, they are not, because they do not satisfy the definition of the term architecture. They more or less just suggests some alternations how to treat addressing and naming differently on the current Internet, thus being nothing else than a band-aid.

In opposite to this, RINA takes all the pieces, which are part of computer communication (as mentioned in Subchapter 2.1, and reassembles them into a new fundamental model of real Internet architecture. Instead of rigid TCP/IP or OSI-RM hierarchical stack of layers with disjunctive functions, RINA postulates the existence of only one general layer with all mechanisms, principles and functions that could be recursively stacked as needed. Besides that, RINA perceives the existence of only two separate protocols that could be used for interprocess communication between RINA layers or applications. The first protocol controls and manages layer, the second one is for data transfer.

RINA is explored and described more in Chapter 5.

3.5.3 Proposals Comparison

The following table Tab. 8 summarizes properties of each proposal above. Abbreviations used as columns names mean:

- *type* – Whether proposal employs map-and-encap (“M”), rewrite (“R”), host-based principle (“H”) or it is something inherently different (“diff”);
- *CE* – Whether proposal is Core-Edge Separation (“CES”), Core-Edge Elimination (“CEE”) or generally different (“diff”) solutions;
- *IPv* = *Internet Protocol version* – Which IP version does proposal supports (“v4/v6/v4v6”);
- *RS* = *Routing Scalability* – Whether proposal reduces DFZ routing tables sizes (“yes/no”);
- *DIL* = *Decoupling of Identification and Localization* – Whether proposal performs (“yes”) locator/identifier split or not (“no”);
- *MH* = *Multihoming* – Whether proposal supports better multihoming or not (“yes/no”), or it is supported conditionally together with utilization of multipath transport protocol (“cond”);
- *Mob* = *Mobility* – Whether proposal supports seamless mobility or not (“yes/no”), or it is supported conditionally together with utilization of multipath transport protocol (“cond”);
- *TE* = *Traffic Engineering* – Whether proposal contains TE by design or not (“yes/no”), or it is supported conditionally with utilization of multipath transport protocol (“cond”);
- *Ren* = *Renumbering* – Whether proposal supports easier renumbering (“yes/no”);
- *Dep* = *Deployability* – Whether proposal allows communication between upgraded and non-upgraded devices (“yes/no”) or whether it is not applicable (“n/a”).

Name	type	CE	IPv	RS	DIL	MH	Mob	TE	Ren	Dep
LISP	M	CES	v4v6	yes	yes	yes	yes	yes	yes	yes
HIP	H	CEE	v6	yes	yes	yes	yes	no	yes	no
SHIM6	H	CEE	v6	no	yes	yes	no	no	no	yes
RANGI	H	CEE	v6	yes	yes	yes	yes	yes	yes	yes
Ivip	M	CES	v4v6	yes	yes	yes	yes	yes	yes	yes
hIPv4	diff	diff	v4	yes	yes	cond	cond	cond	yes	no
NOL	R	diff	v4v6	yes	yes	yes	yes	yes	no	no
GLI-Split	R	CEE	v6	yes	yes	yes	yes	yes	yes	yes
TIDR	M	CES	v4v6	no	yes	yes	no	yes	yes	yes
ILNP	R	CEE	v6	yes	yes	yes	yes	yes	yes	yes
Evolution	diff	diff	v4v6	yes	no	no	no	no	no	n/a
NBS	diff	CEE	v4v6	yes	yes	cond	cond	cond	no	no
APT	M	CES	v4v6	yes	yes	yes	yes	yes	yes	yes
IRON-RANGER	M	CES	v4v6	yes	yes	yes	yes	yes	yes	yes
TRRP	M	CES	v4v6	yes	no	yes	no	yes	no	yes
Six/One	R	CES	v6	yes	yes	yes	no	no	yes	yes
RINA	diff	diff	v4v6	yes	yes	yes	yes	yes	yes	yes

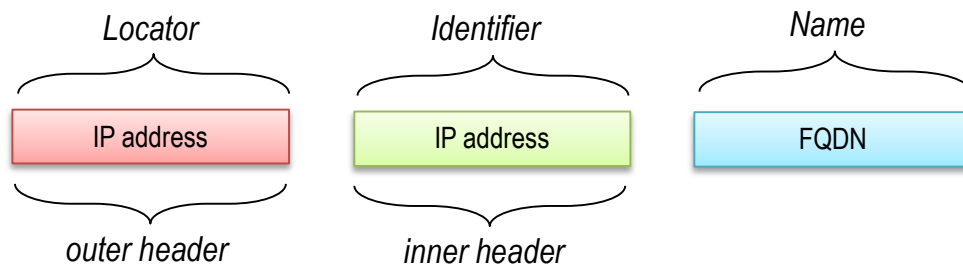
Tab. 8: Properties comparison of existing proposals

Let us focus on comparing CES and CEE solutions because they are a majority of proposals. CES are believed to be superior to CEE, and subsequent paragraphs provide some overview about pros and cons of both.

Main CES features are summarized in the following list:

- Locator/Identifier split is commonly performed as depicted in the Fig. 19;
- Edge networks are separated from DFZ routing tables or are at least highly aggregated. Routing scalability is visible in direct proportion to how widely is CES solution adopted;
- CES benefits are available immediately to adopters – multihoming, inbound TE and if possible also mobility;
- Deployment of CES does not affect DFZ routers, but new devices on the border between core and edge are needed to interconnect this two address spaces together with mapping system;
- CES solutions do not require host stack, API or application changes;
- Tunneling and overlaying impose additional size overhead on fragments, thus introducing MTU concerns when employing CES.

LISP, APT, Ipvip, IRON-RANGER, TRRP:



Six/One:

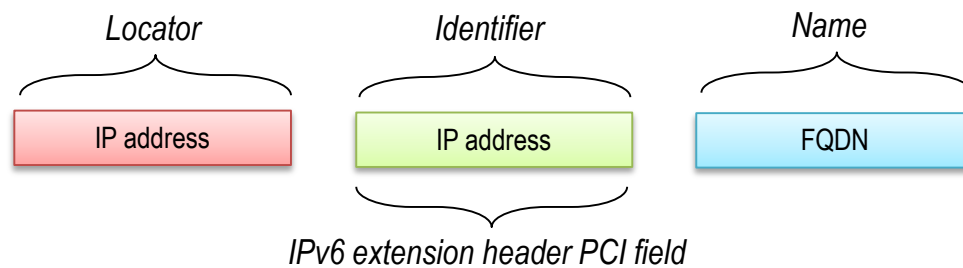
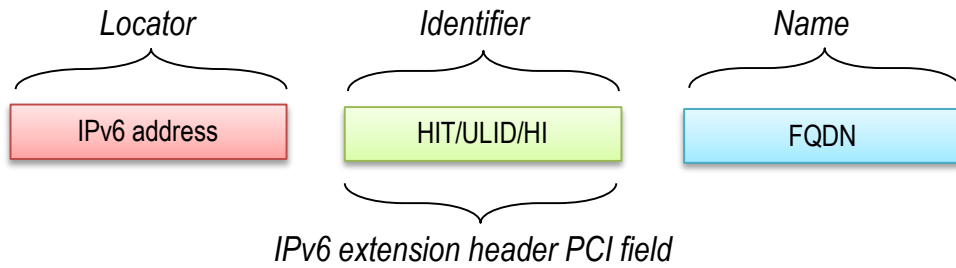


Fig. 19: CES kinds

Main CEE features are summarized in the list below:

- The most of CEE solutions separates locators and identifiers into two completely different *namespaces*. Some representatives are depicted in Fig. 20;
- CEE benefits are visible and widely available to adopters only after majority of network migrate;
- Routing scalability is attained in a way that applications are no longer dependent on stable PI (or de-aggregated PA) addresses. Hence, PA addresses could be easily preferred and administratively more available than PI addresses.
- CEE host stack must determine which locator should use. Besides that, potential set of locators could be retrieved, thus implying resolving multihoming, inbound TE issues, and ideally mobility issues;
- DFZ routers are not affected, and no additional tunneling devices are needed, however, a new infrastructure (or at least upgrade of current one, i.e. DNS) must be present to provide mapping between identifiers and locators;
- CEE solutions need host stack changes and applications augmentations;
- The most of CEE solutions do not support IPv4 and have some troubles with NAT so additionally clutches are needed.

HIP, Shim6, RANGI:



ILNP, GLI-Split:

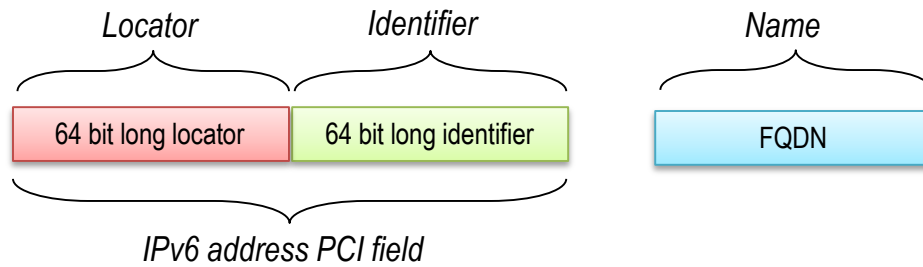


Fig. 20: CEE kinds

It is assumed that CES are easier for voluntarily adoption rather than CEE. On the one hand, the purpose of the routing system is to serve hosts. Hence, the goal is to make routing system more scalable with the help of CES solution that targets network, not hosts. On the other hand, CEE solutions are believed to lead to better final shape of the Internet, because of: a) routing should be as simple as possible without unnecessary tunneling clutches; b) utilization of IP address as identifier is a fundamentally wrong concept. One can say that CES is “network-centric” and CEE is “host-centric”. Unfortunately, no hybrid solution between CES and CEE does exist.

Both of them need a scalable mapping system. Nevertheless, CES mapping system is arguably more efficient because: a) CES lookups are needed only for initial communication towards a host inside edge network in opposite to CEE lookups that must be performed by senders and receivers for any newly established communications; b) CES mapping system is better designed for caching to alleviate unnecessary resolutions; c) it is unlikely that organizations already using PI addresses would downgrade for PA addresses.

3.6 Chapter Summary

This chapter offered theoretical background on naming, addressing, and routing issues. We took into account analogies from other communication sectors to capture invariances and find similarities. We postulated complete naming and addressing model based on a synthesis of important works in this field. Employing previous, we showed contradictions just as in current TCP/IP stack, together with existing band-aids (i.e., IPv6, DNS, and URI).

In the last Subchapter 3.5, we discussed possible solutions. Before anything else, we outlined ideal solution properties and organized their goals according to the importance and beneficial effect. We enumerated existing suitable candidates and briefly mention their specifics. Then we compared and categorized all possible solutions.

Ideal solution should have following properties:

- provide complete naming architecture with one or more levels of addressing indirection, where employed addresses are *location dependent* but *route independent*;
- inherently support use-cases like network multihoming, device's mobility, and owner regulated traffic engineering.

Drawing on overall results and findings, we decided to pursue LISP and RINA more closely to see whether they comply with postulated naming and addressing model and at the same time fit to achieve the most of the ideal solution goals.

4 Locator/ID Separation Protocol

- ☞ *–“Perhaps it’s impossible to wear an identity without becoming what you pretend to be.” O.S.Card*
- ☞ *What is LISP? What components, messages and function does LISP employ?*
- ☞ *Where and how should be LISP used? What is technology readiness level of LISP?*
- ☞ *Can we improve LISP’s operation?*

LISP is currently one of the most discussed Core-Edge Separation solutions that could bring alleviation to “pain points” of nowadays Internet, such as mobility, multihoming, decoupling identity and locality. LISP introduces map and encapsulation technique (map-and-encap) that enables it to be transparent to end-devices and non-LISP network areas. The map and encapsulate principle benefiting from own mapping system to distribute information about identifier-locator pairs. Separation of device identification from its location information is the LISP receipt to the mentioned Internet issues. While the identification of the device should remain constant which is important for addressing network applications, the location information may change depending on the actual position of the node on the Internet. The scalable mapping mechanism is necessary for LISP to work efficiently. The significant research effort was spent on proposing various algorithms for mapping identifiers to locators. These algorithms are discussed and evaluated in this chapter. The second component of LISP’s core principle is encapsulation. The encapsulation takes place at domain borders when the packet needs to be sent outside the local domain. In this case, the packet is encapsulated within a new packet, which header is filled with target address obtained from mapping identifier of the target device to its actual locator address. The advantage of the map-and-encap approach is that it does not require host changes or changes to the core routing infrastructure.

LISP development started after IAB Workshop in 2006, and it supposes to be the response dealing with major problems introduced in Subchapter 2.2. LISP should reduce DFZ routing table growth, stop prefix deaggregation, allow easier multihoming and mobility without the BGP and split locator and identifier namespaces. LISP should be deployed without any changes to hosts or DNS. It must support both IPv4 and IPv6 seamlessly. Moreover, it is agnostic to any network protocol (it could be used with future IPv7 or any new invention working on this layer). Transition mechanisms are part of LISP protocol standard. Thus, it supports communication with the legacy non-LISP world. Nevertheless, the enterprise is always skeptical and slow when adopting new technology. Hence, it is a significant research challenge to investigate LISP features using modeling and simulation as the referential testbed tools producing meaningful outcomes.

In this chapter, we would like to dive into the LISP and explore its capabilities and limitations. The main goal of this chapter is: a) to provide an in-depth presentation of LISP; b) to illustrate known

LISP issues; c) to propose improvements and implement them in the form of new simulation models for OMNeT++; and d) to evaluate the impact of suggested improvements.

4.1 Overview

Majority of this subchapter is based on RFC 6830-6834 [98], [99], [100], [101] that standardize LISP protocol and its interfaces as experimental.

The initial idea behind Internet was to create a simple decentralized connectionless packet switching network that could survive the unpredictable outage of its nodes. From straightforward TCP/IP stack as it was enacted thirty years ago, we moved towards layered model with a variety of “hacks” like MPLS⁶⁶, GRE, IPsec, PPTP⁶⁷, MPTCP that are adding desired functionality but diverting from the original idea, where each layer is present only once, and its function is not repeated. Does this seem like a “simple networking architecture”? IP address functionality is nowadays overloaded as it is explained in Section 2.2.2; it serves both localization and identification purposes. The consequence of this overloading is the inability to build scalable and long-term effective DFZ routing system.

The main idea behind LISP is to separate localization and identification. Following the example of GSM network could serve as an analogy for this. Cellphone identifier is a telephone number, and cell phone localizer is operator’s network, which connects the device. If somebody calls the number (“to identify”) then operator’s network searches for particular base transceiver station (“to localize”) with which cell phone is associated right now in order to establish the call. Whenever owner travels with cell phone abroad, cell phone changes also operator’s network (locator). However, callers are still using the same number (identifier) to reach owner despite the fact that locality has changed.

LISP accomplishes similar behavior by splitting the IP address into two *namespaces*:

- **Routing Locator (RLOC)** *namespace* where addresses fulfill their localization purposes by telling where is device connected to the network (red cloud on Fig. 21);
- **Endpoint Identifier (EID)** *namespace* where each device has a unique name that identifies it from each other (green cloud on Fig. 21).

Also a **non-LISP** *namespace* exists (and probably always will exist), where direct LISP communication is (even intentionally) not supported (blue cloud on Fig. 21). Apart from *namespaces* also exist: a) specialized routers performing map-and-encap that interconnects different *namespaces*; b) dedicated devices maintaining mapping system; and c) proxy routers allowing communication between LISP and the non-LISP world.

⁶⁶ Multiprotocol Label Switching (MPLS). For more, see RFC 3031.

⁶⁷ Point-to-Point Tunneling Protocol (PPTP). For more, see RFC 2637.

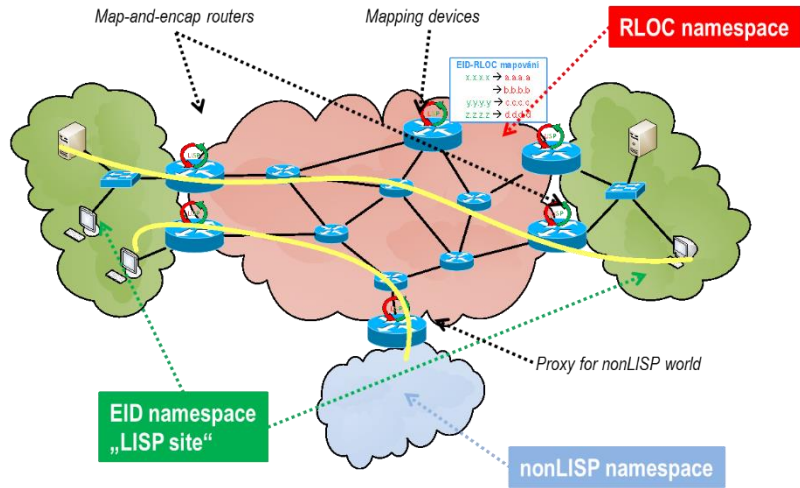


Fig. 21: Basic LISP scheme

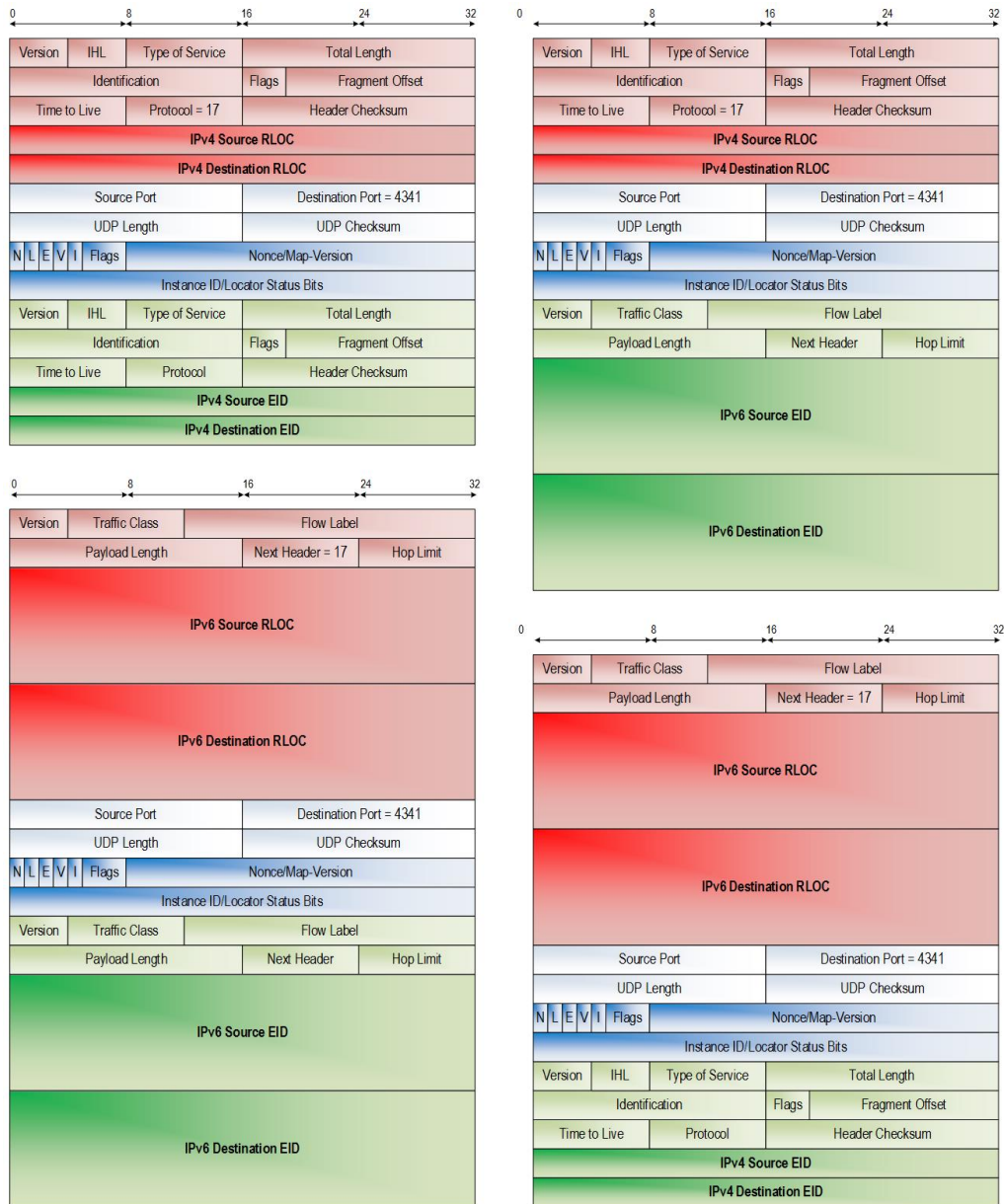


Fig. 22: LISP packet variants

4.1.1 Tunneling

A LISP mapping system performs lookups to retrieve a set of RLOCs for a given EID. Tunnel routers between *namespaces* utilize these EID-to-RLOC mappings to perform map-and-encapsulation. The original (inner) header (with EIDs as addresses) is encapsulated by a new (outer) header (with RLOCs as addresses), which is appended when crossing borders from EID to RLOC *namespace*. Whenever a packet is crossing back from RLOC to EID *namespace*, the packet is decapsulated by stripping outer header off.

LISP supports both IPv4 and IPv6. Moreover, LISP is agnostic to address family thus it can seamlessly work with any future network protocol. Transition mechanisms are part of the protocol standard. Hence, LISP supports communication with the legacy non-LISP world. LISP places between inner and outer header additional PCI in the form of UDP header succeeded by LISP header. LISP uses reserved port numbers – 4341 for data and 4342 for signalization. Currently, any combination of IP headers is supported – **IPv4 outer / IPv4 inner**, **IPv4 outer / IPv6 inner**, **IPv6 outer / IPv4 inner**, **IPv6 outer / IPv6 inner**. However, the map-and-encap principle is so generic that LISP could inherently support any network layer protocol. Fig. 22 depicts all variants of LISP packets.

Basic components are **Ingress Tunnel Router (ITR)** and **Egress Tunnel Router (ETR)**. Both are border devices between EID and RLOC space; the only difference is in which direction they operate. The single device could be either ITR-only or ETR-only or ITR and ETR at the same time (thus abbreviation **xTR**).

ITR is the exit point from EID space (a.k.a. **LISP site**) to RLOC space, which encapsulates the original packet. This process may consist of querying mapping system followed by updating local **map-cache** of recently used mappings. Map-cache improves the performance of the system (i.e., EID-to-RLOC mapping pairs are stored for a limited time to reduce signalization overhead).

ETR is the exit from RLOC space to EID space that decapsulates original header. Outer header, auxiliary UDP, and LISP headers are stripped off. ETR is also announcing all LISP sites (their EID addresses) and by which RLOCs they are accessible.

If we inspect structure of LISP packet somewhere in RLOC space then:

- Inner header source IP = sender's EID address;
- Inner header destination IP = receiver's EID address;
- Outer header source IP = ITR's RLOC address;
- Outer header destination IP = ETR's RLOC address.

4.1.2 Mapping System

Before moving to LISP mapping system concretely, let us discuss how those things are handled theoretically. Any Internet mapping system is nothing else than the huge distributed database. Simple

mapping information is represented in a single database record. We have ended up with two diametrically different approaches how to operate these kinds of databases:

- *Push model* – Any node in the network has the information, or the information is actively propagated through the network to the node. With this approach, “everyone knows everything”. Clear disadvantages are signalization overhead (the number of messages) and resource consumption (CPU and memory requirements) to maintain shared state. The larger the database is, the more computation power is needed. Among existing push model examples are routing protocols maintaining RIB between routers;
- *Pull model* – Information is available to any node, but only upon solicited request. With this approach, “everyone knows just what is needed”. Disadvantages are the level of indirection where the answer to the querier might be altered, outdated or untrusted. DNS is an example of the pull model. DNS divides the whole *namespace* hierarchically to the tree structure in order to avoid single node against knowing all mappings. Then DNS resolver only needs to know whom to ask to retrieve the authoritative answer.

Both approaches have some advantages. However, disadvantages of push model prevent it to be a scalable and dynamic solution beyond a certain point of system size. To illustrate it, IGP protocols are used at the scale of single AS to guarantee the speed of routing convergence. DNS is common protocol handling even more information than just resolving FQDN to IPv4 and/or IPv6 addresses. LISP specifications based on both models exist – LISP-ALT, LISP-DDT, LISP-DHT (previous three will be explained later in text), LISP-EMACS (see [102]) as pull models, LISP-NERD as push model (see RFC 6837 [103]) and LISP-CONS (see draft [104]) as hybrid push/pull model. However, only the ones based on pull model are implemented and operational.

LISP mapping system is primarily employing two components – **Map Resolver (MR)** and **Map Server (MS)**. Looking for EID-to-RLOC mapping is an analogous process as DNS name resolution (see Fig. 23). In the case of DNS, the host asks its DNS resolver (configured within OS) which IP address belongs to a given FQDN. DNS server responds with a cached answer or delegates the question recursively or iteratively to another DNS server according to the name hierarchy. In the case of LISP, querier is ITR that needs to find out which RLOCs could be used to reach a given EID. ITR has preconfigured MR, which is bothered each time mapping is needed.

Queries performing EID-to-RLOC mapping are data-driven. This behavior means that a new data transfer between LISP sites may require a mapping lookup, which causes that data dispatch is stopped until a mapping is retrieved. This behavior allows LISP to operate a decentralized database of EID-to-RLOC mappings. Replication of whole (potentially large-scale) database is unnecessary because mappings are accessed on-demand, just like as in DNS a host does not need to know complete domain database. Tunnel routers maintain map-cache of recently used mappings to improve the performance of the system.

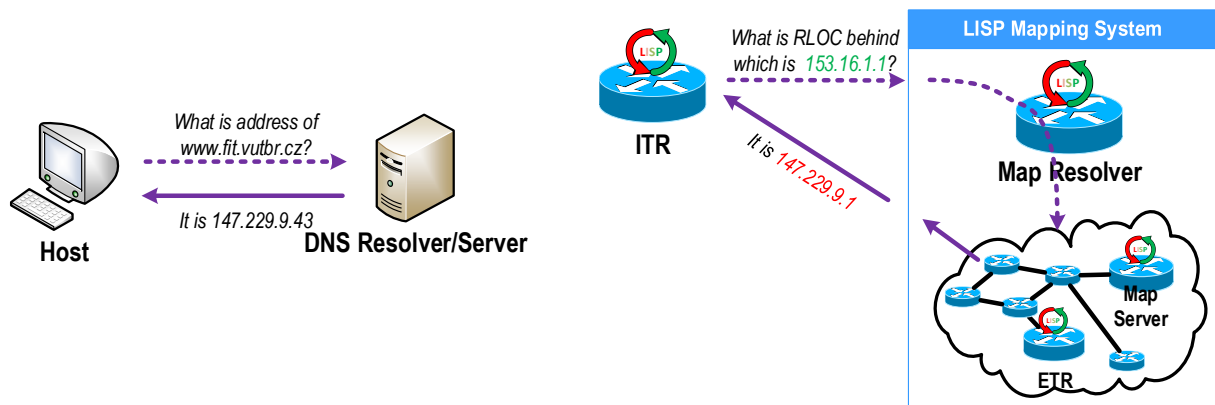


Fig. 23: Comparison between DNS and LISP mapping system

Following list contains all LISP mapping signalization messages with their brief description. They are without inner header – just the outer header, followed by UDP header (with source and destination ports set on 4342), and followed by appropriate LISP message header. Structural details of each message can be seen in Addendum 8.1.

- *LISP Map-Register* – Each ETR announces as authority one or more LISP site(s) to the MS with this message. Each registration contains authentication data and the list of mappings and their properties;
- *LISP Map-Notify* – UDP cannot guarantee message delivery. MS may optionally (when the particular bit is set) confirm reception of *LISP Map-Register* with this message;
- *LISP Map-Request* – ITR generates this request whenever it needs to discover current EID-to-RLOC mapping and sends it preconfigured MR;
- *LISP Map-Reply* – This is solicited a response from the mapping system to a previous request and contains all RLOCs to a certain EID together with their attributes. Each ITR has its map-cache where reply information is stored for a limited time and used locally to reduce signalization overhead of mapping system. Moreover, mapping system generates *LISP Negative Map-Reply* as a response whenever given identifier is not the EID, and thus proxy routing for external LISP communication must occur.

MR processes ITR's *LISP Map-Requests*. Either MR responds with *LISP Negative Map-Reply* if queried address is from a non-LISP world (not EID), or *LISP Map-Requests* is delegated further into a mapping system to appropriate MS.

Every MS maintains **mapping database** of LISP sites that are advertised by *LISP Map-Register* messages. If MS receives *LISP Map-Request* then: either a) MS responds directly to querying ITR (it is allowed to do that because MS has all the necessary information in its mapping database); or b) MS forwards request towards designated ETR that is successfully registered to MS for target EID.

Each RLOC is accompanied by two attributes – priority and weight. **Priority** (one-byte long value in the range from 0 to 255) expresses each RLOC preference. The locator with the lowest priority

is preferred and is going to be used as the outer header address. Priority value 255 means that the locator must not be used for traffic forwarding. Incoming communication may be load-balanced based on the **weight** value (in the range from 0 to 100) between multiple RLOCs sharing the same priority. Zero weight means that RLOC usage for load-balancing depends on ITR preferences.

xTRs perform **RLOC probing** (checking of non-local locator liveness) to always use current information. RLOC probing is done with the help of special variant *LISP Map-Request* and *LISP Map-Reply* messages (with the appropriate bit set on). Let us called them *LISP Map-Request Probe* and *LISP Map-Reply Probe*.

ETR registers itself only to a limited number of MSs. It is technically impossible for all ETRs to be registered to the same MS. Hence, there must be a way how to distribute mapping database and interconnect different MS between each other in order to guarantee the availability of mapping information to all MRs. Following three approaches are the most common:

- **Alternative Topology (LISP-ALT)** – MS are connected via dedicated GRE tunnels across the non-LISP world. LISP routing information are carried as external routes redistributed into BGP. LISP-ALT aggregates EID prefixes and enforces allocation policy. LISP-ALT is not a scalable solution when the number of MSs starts to increase. However, LISP-ALT copes easily with situations when EID identifier blocks are not assigned hierarchically. Fig. 24 depicts three LISP sites exchanging routing information via three dedicated GRE tunnels across the non-LISP core). For more, see RFC 6836 [105].

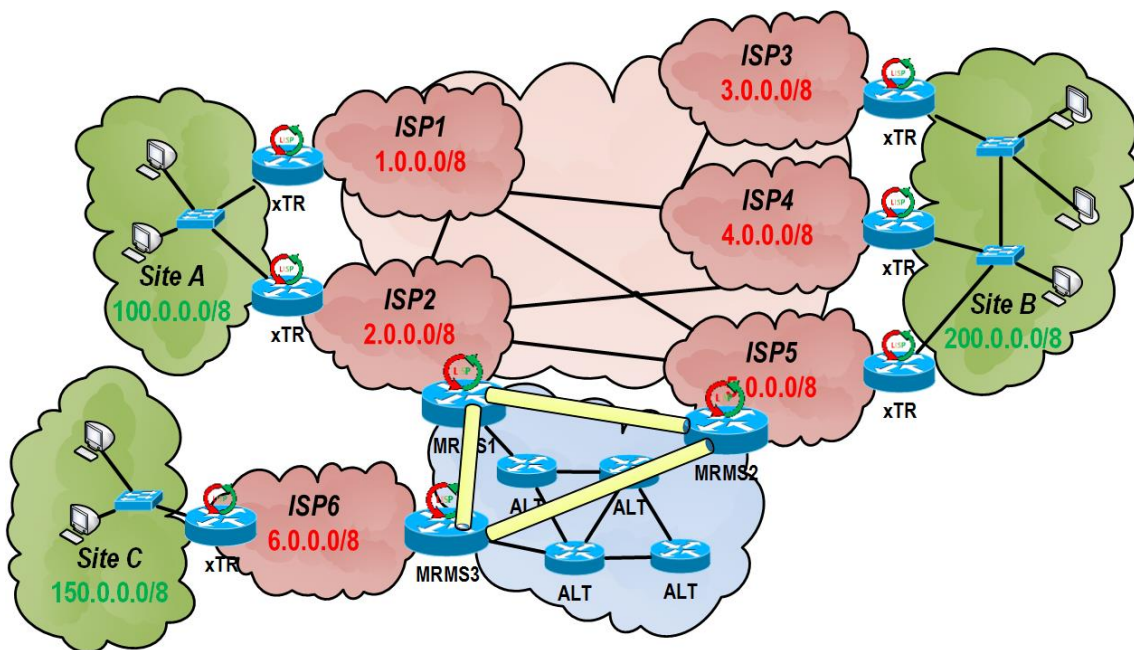


Fig. 24: LISP-ALT infrastructure example

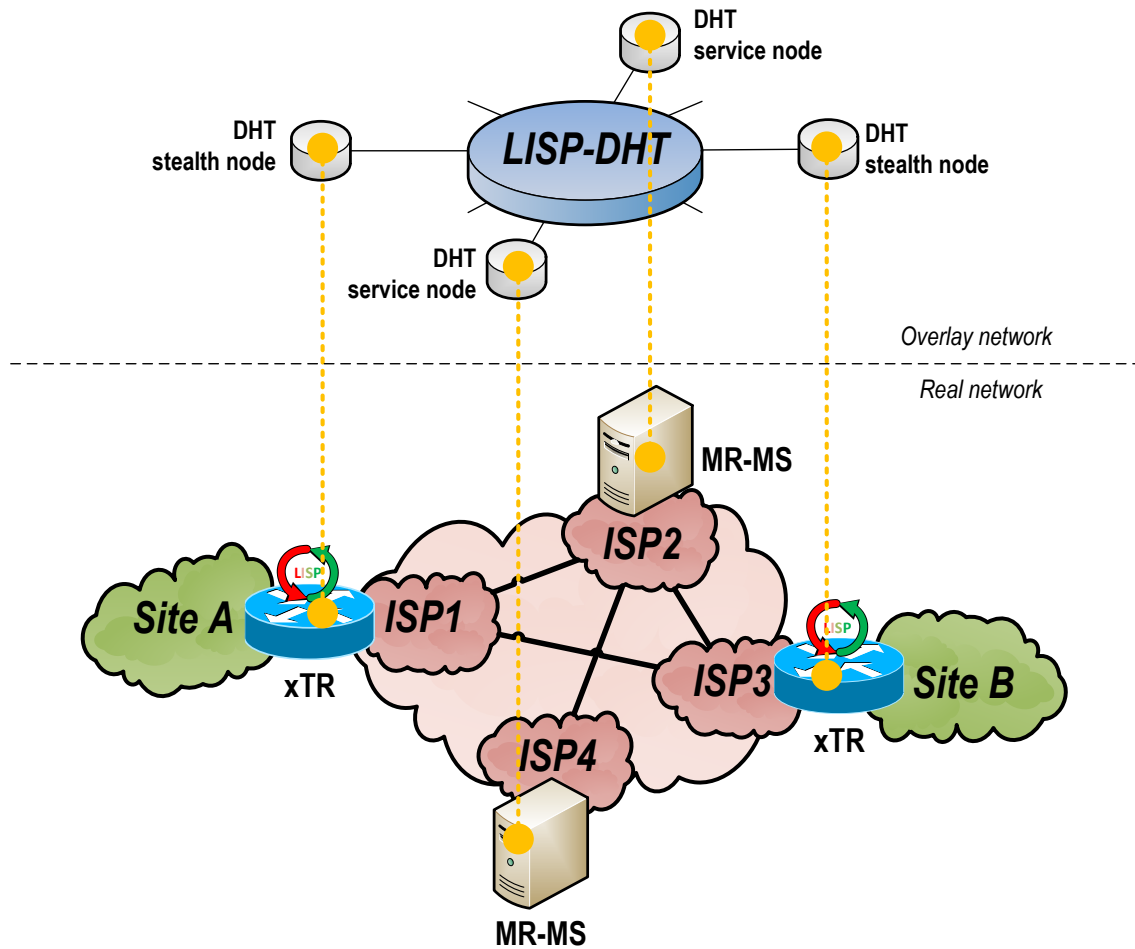


Fig. 26: LISP-DHT infrastructure example

4.1.3 Coexistence between LISP and Non-LISP

Flag Day is not an option in case of migration to LISP just as in the case of IPv6. Moreover, there will always be networks that do not intend to deploy LISP or where LISP deployment is not beneficiary or possible. Special devices are needed to interconnect LISP and a non-LISP world where IP address locality and identity are not decoupled. Communication between those two worlds differs according to the direction, how IP addresses are interpreted during routing procedure and what issues are connected with it:

- *non-LISP* \Rightarrow *LISP* – Hosts and routers do not know anything about loc/id split. Hence, EIDs are considered as ordinary addresses and natively routed to “EID network entry point”;
- *LISP* \Rightarrow *non-LISP* – ITR must recognize that the destination address is not EID. Hence, there are no RLOCs associated with it. The packet is then delivered to “LISP world exit point”.

Two approaches are proposed for LISP/non-LISP coexistence purposes: a) address translation; b) proxies providing ITR and ETR roles (both briefly documented bellow and in [99]).

No matter whether a) or b) is used, the both of them supports Day 1 benefits so that the number of adopters does not determine overall functionality and quality of LISP deployment. Therefore, site profits from LISP (i.e. easier mobility or multihoming) immediately after migration. Full control over inbound TE is the most noticeable adoption gain because of priority and weight attributes that are mandatory to follow by any LISP implementation. Compare LISP load-balancing (according to priority/weight – integral parts of LISP protocol design) and BGP policies that should accomplish the same goal. Unfortunately, BGP policies cannot be enforced and are prone to reconfiguration when traversing ASes.

LISP Network Address Translation

LISP Network Address Translation (LISP-NAT) employs the same principle as classical NAT, which means that xTR translates from EID to global routing prefix and back. A typical use-case is for LISP \Rightarrow non-LISP communication or for LISP sites using same EIDs (e.g. RFC 1918 private addresses as EIDs). This approach is not widely deployed. However, it is easier to integrate it to the control plane of active network devices.

Proxy Ingress and Egress Tunnel Routers

This migration idea is built over proxies that provide ITR and ETR functionality to non-LISP hosts and routers. Two new devices are introduced to LISP architecture – **Proxy Ingress Tunnel Router (PITR)** and **Proxy Egress Tunnel Router (PETR)**.

PITR provides non-LISP \Rightarrow LISP communication, and its goal is to help non-LISP users reach LISP sites. PITR announces highly aggregated EID prefix via routing protocols to the non-LISP world in order to lure and route traffic destined for LISP sites. LISP outer header is wrapped around original data upon sending it via one of the PITR's RLOC interfaces.

PETR provides LISP \Rightarrow non-LISP communication anytime mapping system returns *LISP Negative Map-Reply* as the answer. In this case, the data receiver is non-LISP, and PETR primary serves as a gateway to the non-LISP world. Secondary PETR's objective is to provide communication between LISP sites using different RLOCs address families (e.g. one site is IPv4 and another IPv6).

As in the case of ITR and ETR, the PITR and PETR roles may be delivered dually by a single device called **PxTR**. If communication between hosts goes via two non-dual PITR and PETR then unicast RPF principle might be broken. Therefore, ETR is ignoring unicast RPF checks to prevent any traffic lost.

4.2 LISP Demonstrations

Following demonstrations should help the reader to get more familiar with LISP data traffic and various signaling processes. Each one begins with network graph description, step-by-step walkthrough of each relevant phase accompanied by a picture. Numbers in pictures (the black digit in a yellow hexagon) and walkthroughs (numbered list item) correspond.

4.2.1 Unicast Communication

Fig. 27 depicts two LISP sites (*Site A* using EID prefix 100.0.0.0/24 and *Site B* with prefix 200.0.0.0/24) that are interconnected via RLOC space composed of five ISP networks. *PC-A* with address 100.0.0.99 wants to unicast some data to *PC-B* with address 200.0.0.99. EIDs are transparent from the perspective of hosts; they do not concern about LISP routing.

- #1) Typically DNS query may proceed any IP communication. In the case of LISP, DNS resolver returns EID as IP address associated with *PC-B*'s. DNS A record holds IPv4 EID; DNS AAAA record holds IPv6 EID (e.g. `pc.siteb.com A 200.0.0.99`);
- #2) The packet traverses *Site A* until it reaches *xTR-A2* employing usual IGP routing. *xTR-A2* acts as ITR and prepares appropriate outer header. RLOC is looked up in map-cache based on destination EID 200.0.0.99. Each locator in map-cache has two attributes – priority and weight – where both serve for load-balancing purposes. In case of above demonstration, RLOC 4.0.0.1 is chosen because of the lowest priority;
- #3) Packet traverses RLOC space with 2.0.0.1 as the source address and 4.0.0.1 as the destination address in the outer header (employing locators). The inner header contains 100.0.0.99 as the source address and 200.0.0.99 as the destination address (employing identifiers). Encapsulation of headers is just as same as depicted in Fig. 22, outer header uses protocol number 17, UDP destination port is set on 4341 (reserved value for LISP data);
- #4) The packet is routed via ISPs until it reaches *xTR-B2*'s interface with address 4.0.0.1. This router performs decapsulation (stripping off outer plus auxiliary UDP and LISP headers) and forwards packet to *Site B* based on destination EID address;
- #5) The packet is delivered to *PC-B* having the same structure (single IP header, EIDs as addresses) as it was in #1. LISP functionality is transparent for end-systems, which means there is no need to install or update network stacks or perform additional configurations within OS.

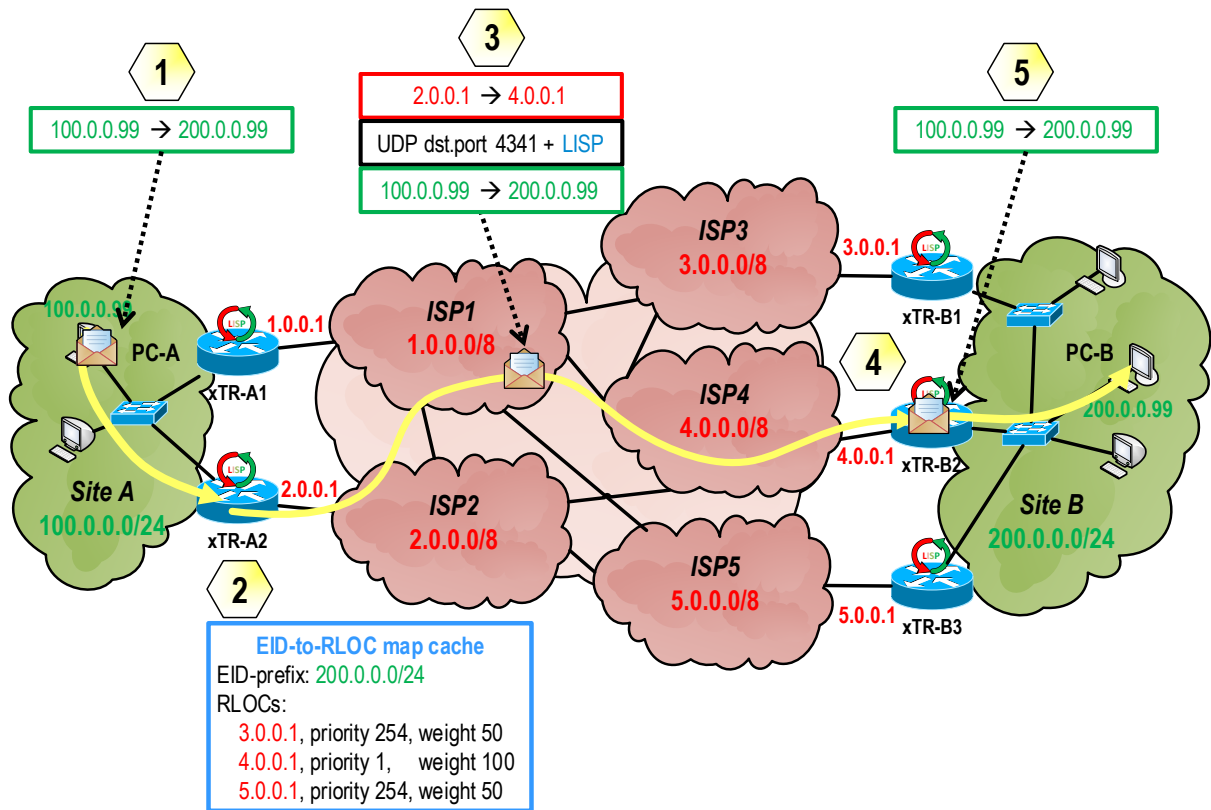


Fig. 27: Illustrative LISP unicast data transfer

4.2.2 Registration

Three routers (*ETR-B1*, *ETR-B2*, and *ETR-B3*) connect *Site B* with RLOC namespace where each one uses different ISP and locator (3.0.0.1, 4.0.0.1 and 5.0.0.1). Available MS are connected via dedicated tunnels employing LISP-ALT. Registration example for *MRMS-B*'s mapping database is shown in Fig. 28.

- #1) ETR periodically generates (by default every 60 seconds) *LISP Map-Registration* message to its preconfigured MS. This message contains EID-prefix and all belonging locators with status vector expressing locators current availability (1 means up, 0 means down);
- #2) *LISP Map-Registration* is delivered to MS where it is processed. Every message implicitly contains an SHA-1 hash of the pre-shared password to protect control plane and provide authentication. Information from message updates existing or creates a new record in mapping database;
- #3) If LISP-ALT is deployed then routing information (existence of successfully registered LISP site) are propagated between *MRMS-A* and *MRMS-B* as redistributed BGP routing updates through GRE tunnel across the non-LISP network. In this scenario, *MRMS-B* announces EID prefix 200.0.0.0/24 to lure traffic intended for *Site B*.

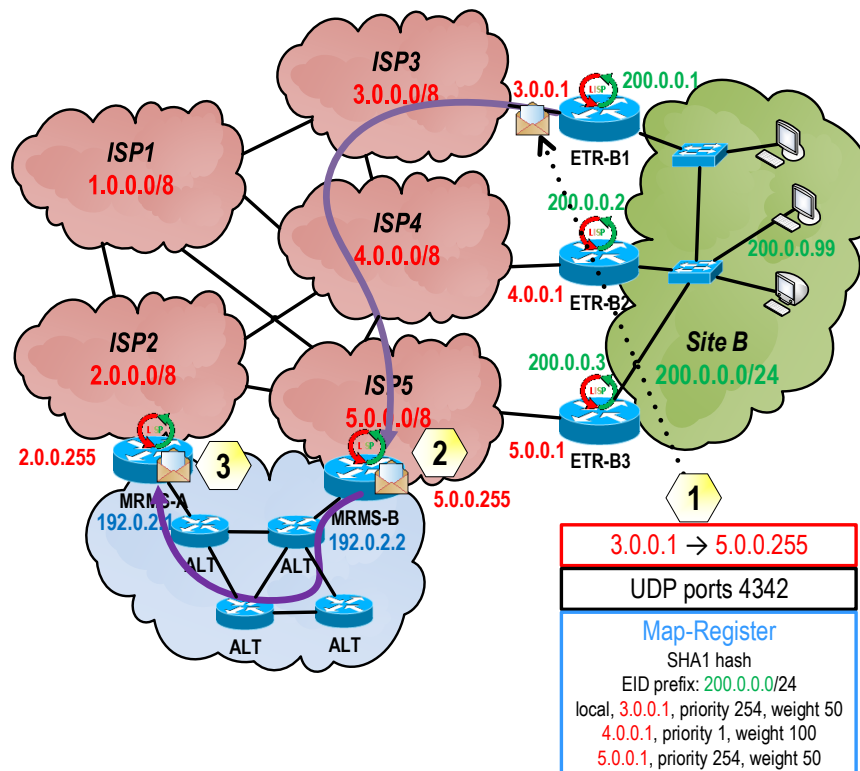


Fig. 28: Illustrative LISP registration process

4.2.3 Mapping Request

Let us revisit scenario of unicast data transfer where a computer with EID 100.0.0.99 wants to communicate with EID 200.0.0.99. The difference is that now *ITR-A2* does not have mapped in its mapping cache. Therefore, *ITR-A2* initializes mapping query to obtain a current set of locators which is illustrated in Fig. 29.

#1) Data traffic drives the generation of *ITR-A1*'s LISP Map-Request message. *ITR-A1* (outer header source address 2.0.0.1) sends a request to its preconfigured MR (inner header destination address 2.0.0.255), where the inner header contains EID addresses of *ITR-A2* (100.0.0.99) and recipient's computer (200.0.0.99). The message body is more complicated than what is depicted in Fig. 29, *LISP Map-Request* contains among others:

- *Nonce* that must be repeated in mapping replies, and that serves as a control plane protection against unsolicited response
- Original sender's address (i.e., 100.0.0.99);
- Input EID list that allows to ask for more than one identifier in a single query (i.e., 200.0.0.99/32);
- RLOC caching data for ETR that answers the request to speed optionally up process (i.e., to EID prefix 100.0.0.0/24 are available locators 1.0.0.1 and 2.0.0.1);

#2) *MRMS-A* accepts mapping request. Subsequently it strips off the outer header and is concerned only with the routing decision based on inner header destination address 200.0.0.99. According

to the routing table, the packet is forwarded through the LISP-ALT tunnel to *MRMS-B* from source address 192.0.2.1 to destination 192.0.2.2.

#3) *MRMS-B* receives *LISP Map-Request* and following next *MRMS-B* lookups its mapping database for ETR that registered requested input EID list item. *LISP Map-Request* is then delegated to one of the registrars, in demonstration scenario to *ETR-B1*.

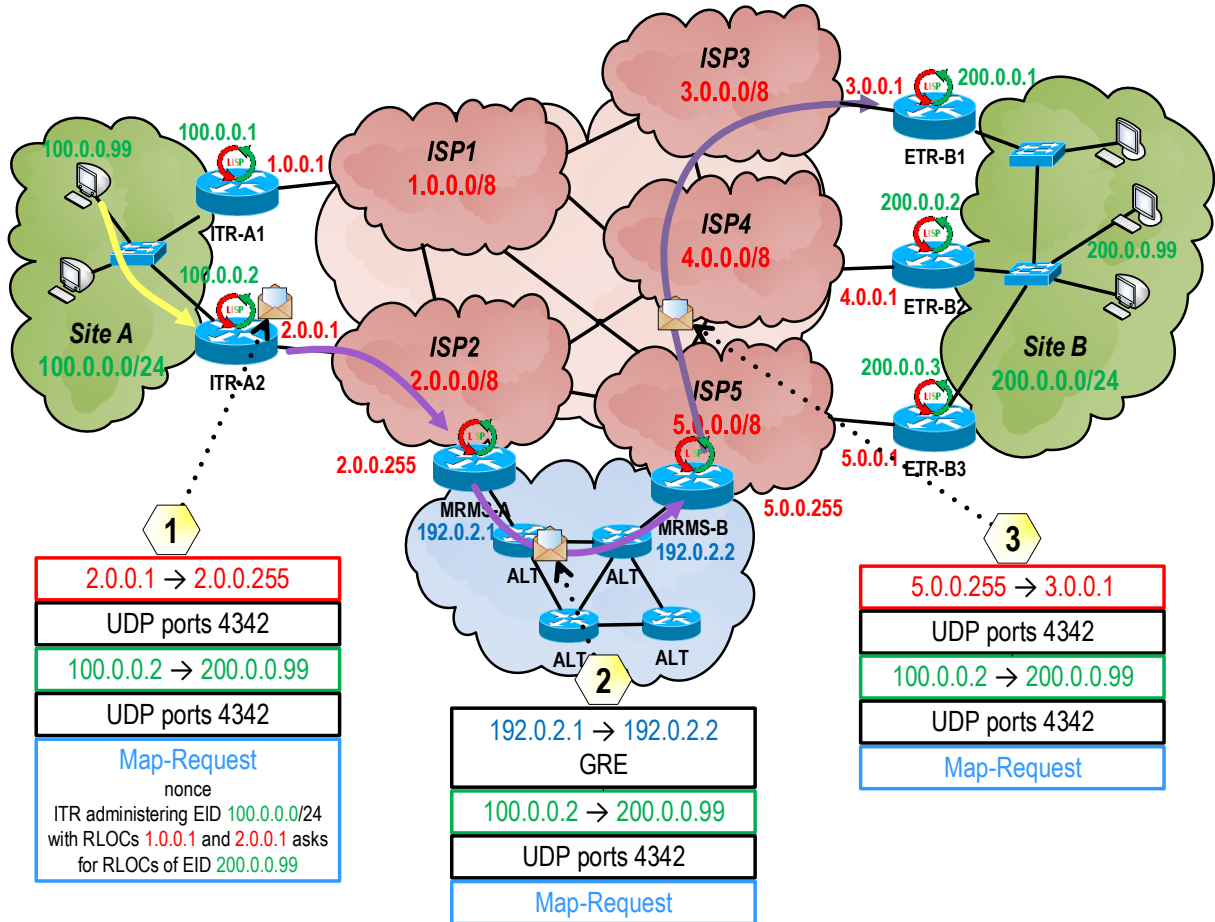


Fig. 29: Illustrative LISP mapping request

4.2.4 Mapping Reply

Two different devices might answer upon receiving mapping request during previously started demonstration scenario.

#1) LISP Map-Reply response message has two ways how it could be generated:

- #a) Either *MRMS-B* previously delegated *LISP Map-Request* to the one ETR registries that answers (it is *ETR-B1* sending it from 3.0.0.1 to 2.0.0.1 in above Fig. 30 labeled as 1a);
- #b) Or registering ETR allows MS to respond to mapping requests instead of ETR with the help of LISP **proxy-reply option** during the registration process. MS responding on behalf of ETR is possible because MS has the same information as ETR in its mapping

database. This option shortens respond delay and overall signaling overhead of protocol that might be appealing for mobile ETRs. In the previous case, it is labeled with option 1b where *MRMS-B* responds with source address 5.0.0.255 towards destination 2.0.0.1.

#2) Sooner or later some response is delivered to *ITR-A2* that initiated mapping query. Upon received, ITR stores current EID-to-RLOC mapping (EID 200.0.0.0/24 could be reached via three RLOCs 3.0.0.1, 4.0.0.1 and 5.0.0.1) into its mapping cache. Finally, unicast communication between PCs 100.0.0.99 and 200.0.0.99 can occur with 4.0.0.1 chosen as locator based on its priority. Data traffic between PCs is discarded (not cached) until mapping query is finished just like as ARP throttling [109]. The previous sentence means that first few packets might be lost during any brand new communication that needs RLOCs that are yet unknown according to the swiftness of mapping response.

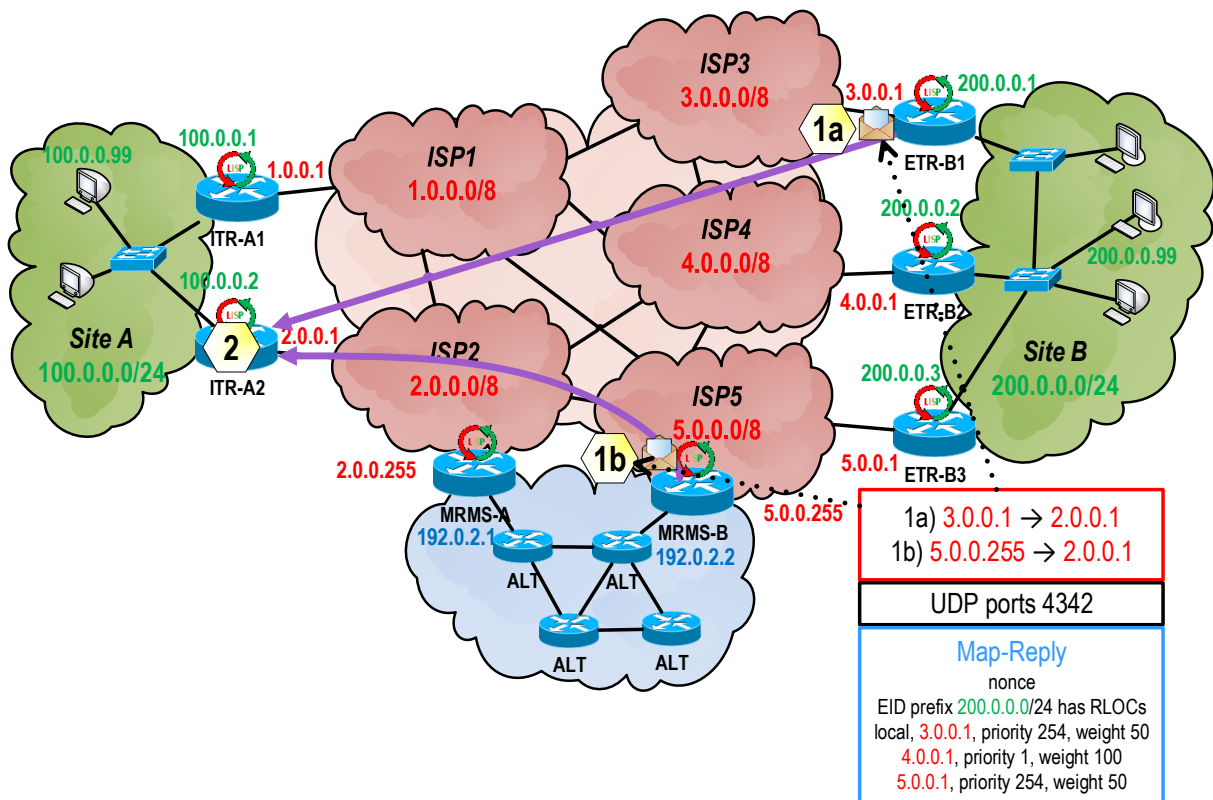


Fig. 30: Illustrative LISP mapping reply

4.2.5 Proxy Communication

Let us focus on bi-directional data transfers between LISP and the non-LISP world. The host with address 9.0.0.99 from the non-LISP world begins communication with PC in *Site B* with address 200.0.0.99. Incoming gateway to LISP world is PITR router that advertises coarsely aggregated prefix 200.0.0.0/8 (depicted with dark green nearby *PITR* on Fig. 31) into which also fits *Site B* EID prefix

200.0.0.0/24. Outgoing gateway from LISP world is PETR device (called *PETR* on Fig. 31) which *Site B* ITRs are using as an intermediate router to pass traffic to the non-LISP world.

- #1) PC with address 9.0.0.99 from non-LISP sends data packet to PC with address 200.0.0.99 in *Site B*. Packet is routed through non-LISP world towards *PITR* because it advertises EID prefixes from LISP world;
- #2) *PITR* wraps the original packet with outer IP header, followed by UDP with destination port 4341 accompanied by LISP header. Outer header has 3.0.0.254 as source and 4.0.0.1 as destination address because it is a locator for destination identifier;
- #3) The packet traverses RLOC namespace until it reaches ETR's locator interface 4.0.0.1. *xTR-B2* removes additional headers and packet are forwarded to *Site B* and to end receiver with address 200.0.0.99;
- #4) Communication is usually bidirectional, hence 200.0.0.99 replies to 9.0.0.99. Classical routing delivers answer to ITR *xTR-B2*;
- #5) ITR performs mapping query to lookup 9.0.0.99. However, mapping system returns *LISP Negative Map-Reply*, which means that destination is not a part of LISP world, and it should be routed via Proxy ETR. Auxiliary headers are added and then the packet is sent towards preconfigured *PETR* with address 1.0.0.254.
- #6) *PETR* decapsulates additional headers and forwards packet towards recipient 9.0.0.99 in the non-LISP world.

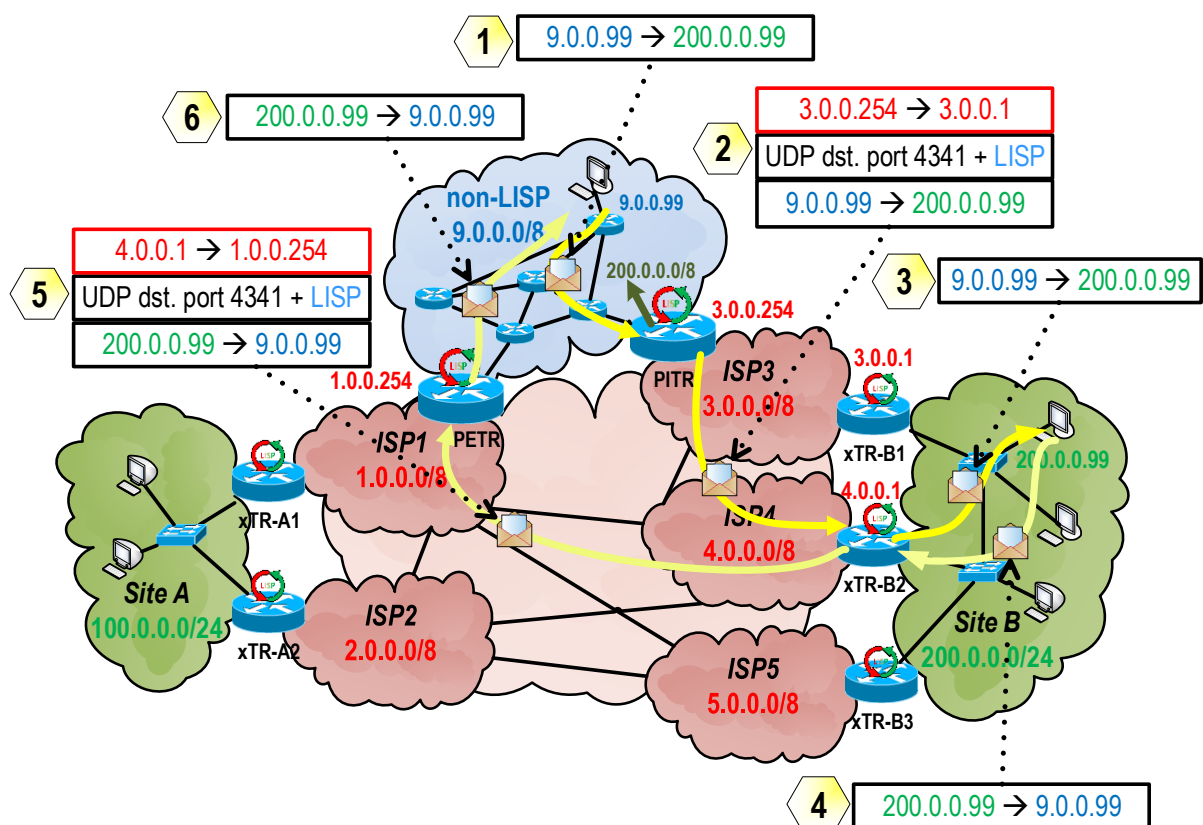


Fig. 31: Illustrative communication between LISP and non-LISP world using PITR and PETR

4.3 State-of-the-Art

This subchapter discusses available LISP implementations for both real and simulated environment. Moreover, it outlines LISP test-bed network and adoption level by the enterprise.

4.3.1 Implementations

Despite the fact that Cisco employees did major LISP protocol design, Cisco does not claim any legal rights. LISP is being further developed within IETF open standardization process and its working group [110]. Hence, more than one referential implementation is available for customers and developers.

OpenLISP

OpenLISP [111] is the first open-source implementation of Unix-based systems, namely for FreeBSD 7 and FreeBSD 8. Unfortunately, it is not being developed anymore. Thus, the latest LISP control plane version is from April 2012, and it does not contain any additional functional improvements.

Cisco IOS/IOS-XR/NX-OS

Operating systems in Cisco devices contains the most up-to-date LISP implementation [112]. LISP is available in relevant releases of:

- IOS since version 15.1 for Catalyst 6000, Cisco 810, Cisco 880, Cisco 890, Cisco 1941, Cisco 2900, Cisco 3900 and Cisco 7200 platforms;
- IOS-XE since version 3.3 for ASR 1000 platform;
- IOS-XR since version 4.3 for ASR 9000 platform;
- NX-OS since version 5.2 for Nexus 7000 platform.

LISPMob

LISPMob [113] is an open-source project which offers LISP control plane and also **LISP mobile node** [114] implementation, which allows devices like smartphones or tablets to benefit from LISP seamless mobility. The project is multiplatform and currently supports all Linux-based systems including Android or OpenWRT.

AVM Fritz!OS

AVM Fritz!OS [115] contains LISP implementation intended for Fritz!Box 7390 platform that offers xTR and MS functionality.

4.3.2 Deployment

LISP BetaNetwork [116] is one of the first project focusing on widespread deployment. Currently, 600+ organizations (among others e.g. Google, Facebook, Cisco, Qualcomm, AT&T, Lufthansa and Microsoft) from more than 34+ countries have joined it voluntarily during its five years existence.

Any organization may request to participate. After the quick review process, IPv4 pool of address from prefix **153.16.0.0/16** and an IPv6 pool of address from prefix **2610:d0::/32** are assigned to the organization. Both prefixes belong to ASN 3943 [117] so that Pitr and Petr routers have own AS from the perspective of the non-LISP world. Other information like assigned MSs, MRs and PxTRs are provided by the registrar as shown for FIT-BUT in Fig. 32.

```
fit-xtr:
Device Type      - {IOS/FreeBSD}
Geographic       - Czech Republic
DNS Name         - fit-xtr
EID-Prefix Set  - {153.16.48.112/28, 2610:D0:214D::/48}
RLOC Set         - {tbd}
Map-Servers     - {RIPE}{l3-london-mr-ms 195.50.116.18   intouch-ams-mr-ms-1 217.8.98.42}
                  - {RIPE}{tdc-mr-ms      193.162.145.50   intouch-ams-mr-ms-2 217.8.98.46}
Map-Resolvers   - {RIPE}{l3-london-mr-ms 195.50.116.18   intouch-ams-mr-ms-1 217.8.98.42}
                  - {RIPE}{tdc-mr-ms      193.162.145.50   intouch-ams-mr-ms-2 217.8.98.46}
PXTR (RIPE)    - {intouch-pxtr-1}{217.8.98.33, 2001:67C:21B4:107::b}
```

Fig. 32: FIT-BUT’s LISP BetaNetwork registration

Global connectivity to LISP BetaNetwork could be verified by: a) **LISP Internet Groper (LIG)** (see RFC 6835 [118]), which is versatile usually command line tool for generating mapping requests capable of retrieving locators to a given identifier; b) tools like LISPmon [119], which can verify successfulness of ETR site registration.

4.3.3 Simulators

The research community has limited options how to observe and expand LISP features in a safe environment of simulator where different scenarios could be easily scheduled and verified later.

One of a few attempts is CoreSim developed by Coras et al. [120]. It is written in Perl, and it allows predict ITR and MS behavior at a macro-scale level using traffic traces, BGP data, and latency estimations. However, CoreSim estimations use rather a general mathematical model taking into account only the distance [121]. Currently, limited LISP implementation exists authored by Hoefling et al. [122] to support LISP MobileNode NAT traversal [123]. However, it is intended for outdated INET-20100323 and OMNeT++ 4.0. Previously, LISP map-cache performance have been evaluated employing high-level simulation that is not taking into account protocol implementation specifics [124].

Among other goals of this thesis is to provide the community with a variety of simulation models supporting up-to-date version of LISP protocol.

4.4 Contribution

LISP architectural implications are discussed in IETF draft [125] followed by companion paper [126]. Previous papers outline and discuss two major issues for LISP threatening its scalability – Site-Based State Synchronization Problem and Locator Path Liveness Problem.

Site-Based Synchronization Problem occurs whenever EID-to-RLOC mappings (including locator statuses) may need to be shared among nodes. Remember that LISP mapping queries are data-driven. There is no need to rediscover mapping for the same data traffic by one xTR if this mapping is already known to other site's xTRs. Sharing of mapping improves routing of packets in case of asymmetrical traffic flows. Imagine that traffic is leaving the site via two xTRs – one is actively dispatching all traffic, another is backing up its functionality. Map-cache on active xTR is populated with records whereabouts map-cache on backup has no mapping state. Whenever traffic shifts from active path to backup path, former backup xTR experiences map-cache misses

Locator Path Liveness Problem is formulated by a question whether given set of source locators and a set of destination locators, can bi-directional connectivity be determined between the $\langle \text{srcRLOC}, \text{dstRLOC} \rangle$ address pairs? Locator Path Liveness Problem is present not only in LISP but its variants also apply to other candidates like HIP, SHIM6 or IRON-RANGER. In the case of LISP, if ITR chooses destination RLOC, which is not reachable, then traffic is discarded somewhere along the path towards destination LISP site.

This subchapter introduces two proposed improvements targeting some of the issues from previously mentioned papers that increase LISP performance – map-cache synchronization and merged RLOC probing. In order to evaluate contribution, we developed brand new OMNeT++ simulation modules for LISP and also for Virtual Router Redundancy Protocol that is being deployed simultaneously on ITR.

4.4.1 Virtual Router Redundancy Protocol

This section briefly outlines Virtual Router Redundancy Protocol because it is closely connected with Site-Based Synchronization Problem scenarios. LISP is being successfully deployed in enterprise networks, and one of its most beneficial use-cases is for data-centers networking. An important feature of any data center is its ability to maintain high-availability of provided services. This feature is accomplished mainly with redundancy. In the case of an outage, service delivery is not affected because of redundant links, devices or power sources. **Virtual Router Redundancy Protocol (VRRP)** is among related protocols and technologies guaranteeing redundancy and helping to achieve high-availability. VRRP is widely adopted protocol providing redundancy of **default-gateway**⁶⁸.

⁶⁸ **Default gateway:** A crucial L3 device that serves as exit/entry point to a given network. For more, see http://en.wikipedia.org/wiki/Default_gateway

VRRP combines redundant first hop routers into virtual groups. One master router actively forwards client's traffic within each group, where others in the group are backing its functionality. Backup routers are periodically checking the liveness of the master waiting ready to substitute it in the case of failure. Switching to a new active router is transparent from the host's perspective thus no additional configuration or special software is needed.

VRRP specification is publicly available as RFC standard – RFC 3768 [127] describes IPv4-only VRRPv2 and RFC 5798 [128] describes dual IPv4+IPv6 VRRPv3. VRRPv2 routers send control messages to multicast address 224.0.0.18. VRRPv3 routers use ff02::12 for IPv6 communication. VRRP has its own reserved IP protocol number 112.

Clustered redundant routers form a VRRP group identified by **Virtual Router ID (VRID)**. Within the group, a single router (called **Master**) is elected based on announced **VRRP priority** (a number in the range from 1 to 255). Higher priority means a superior willingness to become Master, zero priority causes the router to abstain from being Master. In the case of equal priority, binary higher IP address serves as tie-breaker. VRRP election process is always preemptive (unlike to non-preemptive HSRP or GLBP), which means that router with the highest priority always wins to be the Master no matter whether group already have got other Master elected. Only Master actively forwards traffic. Remaining routers (called **Backups**) are just listening and checking for Master's keep-alive messages.

Hosts have configured virtual IP address as their default gateway. Only Master responds to *ARP Requests* for this IP. This IP address has assigned reserved MAC address – 00:00:5e:00:01:\$\$ for VRRPv2 and 00:00:5e:00:02:\$\$ for IPv6 (where \$\$ is VRID). Whenever VRRP group changes to a new Master, *ARP Gratuitous Reply* is generated to rewrite association between an interface and reserved MAC in CAM table(s) of the switch(es). This behavior allows transparent changing of Masters (in the case of an outage) from host's perspective.

VRRP has only one type of control message – *VRRP Advertisement*. If Master is not elected, then VRRP routers exchange advertisements to determine which one is going to be a new Master. If Master is already elected then, only Master is sending *VRRP Advertisements* to inform Backups that it is up and correctly running. *VRRP Advertisement* is generated whenever advertisement timer (*AT*) expires (by default every 1 second). If this interval is set to a lower value then Master's failure is detected faster but protocol overhead increases. Master down interval (*MDI*) resets with each reception of an advertisement message. Backup, which expires the *MDI* sooner, becomes a new Master. Value of *MDI* depends on priority of each VRRP router according to (1). The highest (best) priority Backup times out first (because of the lowest *skew time*) and thus takes over role as a new Master before others.

$$MDI = 3 \times AT + \frac{\overbrace{(256 - priority) \times AT}^{skew\ time}}{256}$$

OMNeT++ VRRP module is a byproduct of this thesis needed for accurate simulation of high-availability scenarios allowing easy forming of active and backup paths for traffic.

4.4.2 Map-Cache Synchronization

Assume multiple redundant routers are acting as first hops in the high-availability scenario like in Fig. 33. Those routers are simultaneously clustered into VRRP groups and act as LISP's xTRs – they run LISP and VRRP at the same time.

The performance of map-and-encap depends on the fact whether xTR's map-cache contains valid EID-to-RLOC mapping or not. Dispatched data traffic drives map-cache record creation. If map-cache misses the mapping, then, a mapping system needs to be asked, and initiating data traffic is meantime dropped. This fact is illustrated in Fig. 33 for EID address $y.y.y.y$. On the one hand, packets (with $y.y.y.y$ as destination) can traverse *ITR1* without any problem (locator $c.c.c.c$ is present in map-cache). On the other hand, same packets are discarded on *ITR2*, which misses the mapping. Packet dropping is a logical step as long as the mapping is not discovered because map-and-encap cannot occur without proper information. The rationale behind this behavior is the same as in the case of ARP throttling [11], where any triggering traffic should be discarded to protect control-plane processing and prevent superfluously recurrent mapping system queries.

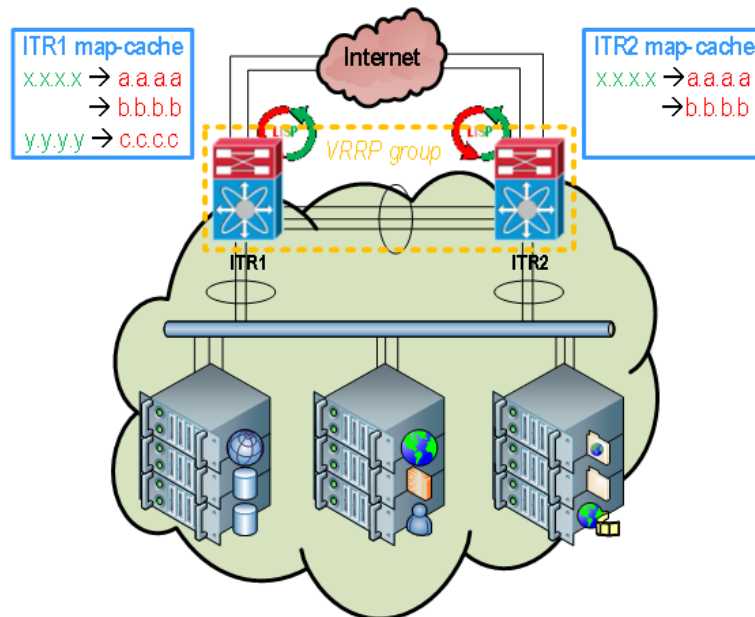


Fig. 33: Site-Based State Synchronization Problem illustration

Each xTR has its map-cache, and its content may differ even within the same LISP site because other traffic may initialize various map-cache entries. Hence, xTRs can easily experience severe packet drops and LISP control message storms due to the map-cache misses when Master change occurs within VRRP group.

Previous is known as **Site-Based State Synchronization Problem**. If we have two or more redundant xTRs, then we want to reduce packet drops as much as possible in case there is a traffic shift from an active to a backup device. xTR outage leads to the off-site signaling storm (lots of LISP Map-Request/Reply messages being exchanged) and dispatching delay for ordinary traffic.

This problem is described as the one of LISP weak-points in [129] and theoretically investigated in [130]. The viable solution would be to provide map-cache content synchronization that should minimize map-cache misses upon failure. Inspired by that, we present our solution addressing this problem.

We have decided to implement it as a technique maintaining synchronized map-caches within a predefined **synchronization set (SS)** of ITRs. Any solicited *LISP Map-Reply* triggers synchronization process among SS members.

SS members are identified and reached using the IP address. Following strategies might be used when choosing appropriate SS member address:

- SS address comes from non-LISP world – Either IP address should be loopback or address of dedicated interconnection shared by all SS members. In the first case, unique device loopbacks need to employ additional routing. In the second case, the additional port for the dedicated connection is seldom available. Also, tracking of SS member needs additional LISP control plane updates;
- SS address comes from LISP world:
 - SS address is RLOC – SS membership is bound to the operability of a given RLOC interface, but this has negative implications for the situation, where xTR has more than one RLOC available. Although, it is easy to track SS member status using return value of RLOC probing;
 - SS address is EID – The best option reflecting LISP’s ideology. EID as SS address should be reachable via direct routing (xTRs share common EID segment) or unless all RLOCs to this EID are down (which could be also used to track peer synchronization status).

Each record in the map-cache is equipped with a **time-to-live (TTL)** parameter. TTL expresses how long the record is considered to be valid and usable for map-and-encap. By default, every record uses the same initial TTL value. Map-caches within SS must maintain the same TTL on shared records; otherwise a loss of synchronization might occur (on some ITRs, identical records could expire because of no demand for traffic).

Either SS membership may be completely stateless, or SS member may maintain a state of its synchronization peers. The stateful approach allows sending of partial synchronization updates. We have implemented two modes of synchronization reflecting previous observation:

- 1) *Naïve* – The whole content of map-cache is transferred to SS. All mappings are then updated according to the new content and TTLs are reset. This approach works fine, but it obviously introduces significant transfer overheads;

- 2) *Smart* – Only record that caused synchronization is transferred. However, peer synchronization status have to be employed to deal with the situation when SS member goes back up and completely lacks any mapping. At that time, a whole set of map-cache content must be sent (not just a partial update). Moreover, we bound this mode with the following policy. When TTL expires, the ITR must check record usage during the last minute (one minute should be a period long enough to detect ongoing communication). If the mapping has not been used (based on the last lookup time of cache record), then it is removed from the cache. Otherwise, its state is refreshed by query followed by synchronization.

Both approaches guarantee that devices within SS could forward rerouted LISP data traffic without packet loss or interruption because they share the same content as ITR's map-cache of malfunctioned former Master.

Synchronization itself is done with the help of two new LISP messages – one carries synchronization data, another optionally acknowledges successful synchronization:

- *LISP CacheSync* – It contains map-cache records, which are being synchronized, and authentication data, which protect SS members from spoofed messages;
- *LISP CacheSync Ack(nowledgement)* – Because LISP leverages UDP, it cannot guarantee message delivery. However, we decided to employ the same principle as for *LISP Map-Register* and *LISP Map-Notify*. Hence, *LISP CacheSync* delivery may be optionally confirmed by echoing back *LISP CacheSync Ack* message.

Message structure of *LISP CacheSync* is depicted in Fig. 34 and *LISP CacheSync Ack* in Fig. 35. Notable differences when comparing to *LISP CacheSync/(Ack)* with the structure of *LISP Map-Register/Notify* are:

- Both messages also include new Type values – *LISP CacheSync* is 5, *LISP CacheSync Acknowledge* is 6;
- *LISP CacheSync* header contains C flag. When C flag is set on, then synchronization acknowledgment is requested by a sender. Receiver (i.e., SS member) must reply with *LISP CacheSync Ack* containing all the map-cache records that have been successfully processed. *LISP CacheSync* message is resent after the acknowledgment awaiting timeout (by default with cumulative value $2^{\text{numOfRetries}}$);
- There is no need for A flag in Cache Record and L and p flags in RLoc (for details about flag meanings, please see [98]);
- As in the case of *LISP Map-Register/Notify*, *LISP CacheSync/(Ack)* mandatorily contain nonce and authentication using HMAC to avoid spoofing of false unsolicited cache synchronization information.

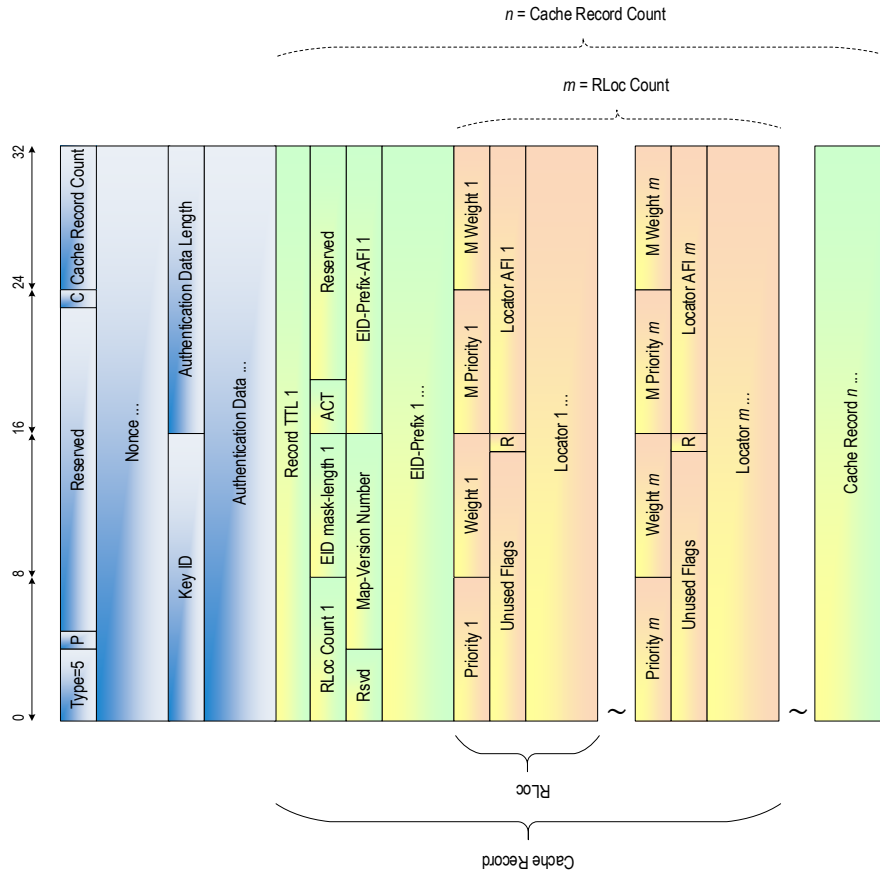


Fig. 34: LISP CacheSync message format

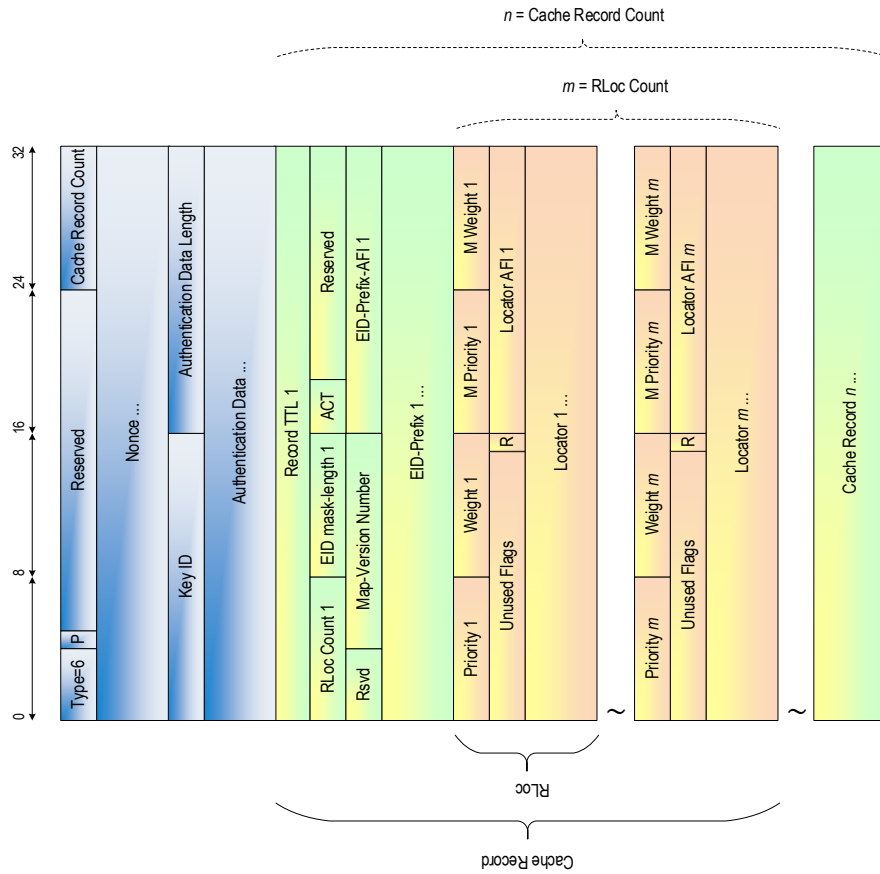


Fig. 35: LISP CacheSync Acknowledgment message format

The diagram in Fig. 36 depicts FSM implementing map-cache synchronization where transitions are denoted with “input / action” labels. Our solution provides a clean-slate way how to alter the content of the map-cache reliably. Nevertheless, others might try to leverage options already available in LISP. Unfortunately, each one has some disadvantage.

The first approach is to alter existing *LISP Map-Requests* by forcing included map-reply record field to contain more than one record. However, this approach is unreliable because it lacks acknowledgment scheme and cannot solve all following wrong goings. What if receiver side does not recognize this option inside *LISP Map-Request*? What if *LISP Map-Request* did not reach receiver? What if the receiver wants to process only part of synchronization information? What if SS-members need to synchronize map-cache when the condition for sending *LISP Map-Request* is not met?

The second approach is that LISP already contains an on-demand renewal of mapping information called **Solicit-Map-Request (SMR)**. SMR is a mechanism how ETRs may rate-limit requests and notify ITRs about mapping change. When mapping changes, ETR starts to send *LISP Map-Request* (with the SMR-bit set on) messages to ITRs with which it recently exchanged data. Then, ITR generates SMR-invoked *LISP Map-Request* to discover new mapping. If we want to use SMR to push new mappings into ITR’s map-cache, then the best way seems to be extending the functionality of MR (see [130]). However, this approach yields significant off-site signalization overhead.

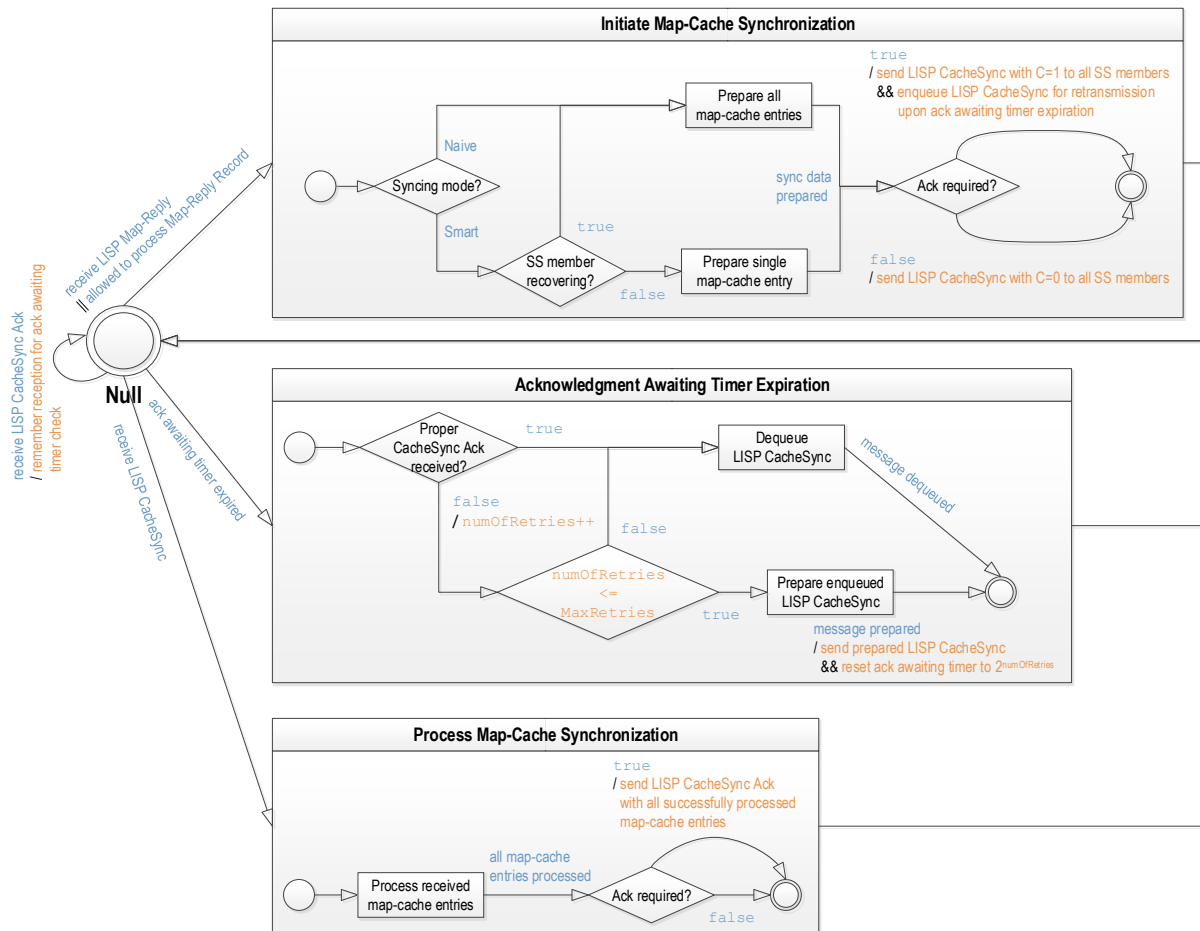


Fig. 36: Map-cache synchronization operation

4.4.3 Merged RLOC Probing

Locator Path Liveness Problem concerns whether a destination locator is reachable via particular source locator or, in other words, whether bi-directional connectivity exists between a given pair of locators. Problem relevant to LISP is depicted in Fig. 37 where *xTR-A1* asks for *Site B* locators. In this case, two locators are available (1.0.0.1 and 2.0.0.1). *xTR-A1* chooses the second one as a destination address for packets. If the link between *ISP1* and *ISP2* goes (un)intentionally down, 2.0.0.1 is not reachable anymore, and *xTR-A1* must somehow find out this fact.

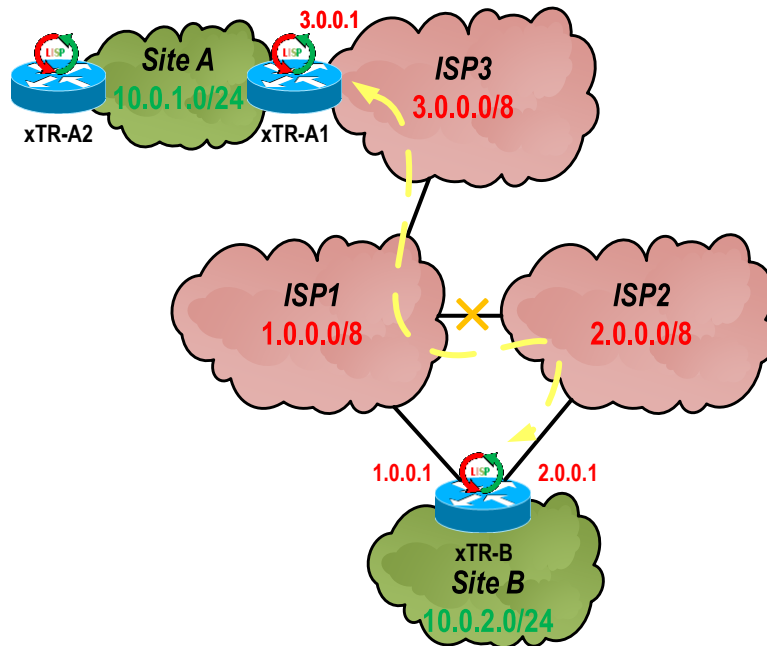


Fig. 37: Locator Path Liveness Problem illustration

Locator Path Liveness detection (checking whether RLOC is reachable or not) does not scale very well in large networks because the reachability of every destination locator must be probed against every source locator of a given device. Complexity of such a task is generally $O(n \times m)$, where n is a number of source and m a number of destination locators. However, instead of brute-force probing some hints might be used to mitigate (but not to avoid) such complexity, e.g. piggybacking, timeouts, existence of underlying routing, positive feedback from protocol control messages or other protocols.

To make Locator Path Liveness Problem even more complicated, let us imagine a situation when LISP site has two or more ITRs with different destination locator reachability. One ITR has connectivity, and another has not (e.g. *xTR-A1* and *xTR-A2* on Fig. 37). Hence, all packets processed by that ITR are going to be discarded somewhere in the network. Unfortunately, neither IGP responsible for routing the packet to faulty ITR nor hosts have capabilities to detect this issue from their subjective point of view.

In order to find a remedy for this problem, we focused on the behavior of Cisco referential implementations and their RLOC-probing algorithm checking locator reachability. ITR is probing assigned locators for each configured EID. This behavior is in compliance with [98] but it leads to repeated check of the same locator multiple times, which represents scalability issue in larger networks.

We decided to decrease protocol overhead by merging EIDs to check locator liveness with a single RLOC probe that we call **merged RLOC probing**.

The *simple* but rather a trivial approach would be to make the following assumption: “If the same locator is reachable for one EID then it would also be reachable for other EID.” Hence, the router can generate only single RLOC probe during one liveness checking period. If it receives positive *LISP Map-Reply Probe*, it may consider probed locator as alive for all EIDs in map-cache that are using it. More *sophisticated* approach is to:

- 1) On sender, check liveness of a given locator with a single *LISP Map-Request Probe* containing one or more query records. Each query record specifies cached EID that uses probed RLOC;
- 2) On receiver, respond with *LISP Map-Reply Probe* that includes locator status updates for all queried EIDs contained in request (or only subset of those EIDs that are in up state);
- 3) Back on the sender, refresh locator status of relevant EIDs in map-cache according to answer(s) in reply.

Above described mechanism is compatible with RFC description and does not need any protocol extensions. It preserves the accuracy of Cisco’s RLOC probing algorithm but with only single RLOC probe exchanged. We have integrated all above described algorithms – *Cisco’s*, *Simple* and *Sophisticated* – in our LISP simulation module.

4.4.4 Design and Implementation

The ANSA project (Automated Network Simulation and Analysis) running at our university is dedicated to developing the variety of simulation models compatible with RFC specifications or referential implementations. Subsequently, these tools allow formal analysis of real networks and their configurations. They may be publicly used as the routing/switching baseline for further research initiatives, i.e., in simulations for proving (or disproving) certain aspects of technologies and/or related protocols. In the frame of this project, we have developed **ANSARouter** as simulation module mimicking behavior of real generic Cisco router.

We have implemented LISP as OMNeT++ compound module called `LISPRouting`, which provides independent xTR, MR, and MS functionality. It consists of five submodules that are depicted in Fig. 38 and described in subsections below the figure. `LISPRouting` exchanges messages with UDP submodule, IPv4 `networkLayer` and IPv6 `networkLayer6` modules of INET framework. `LISPRouting` integration within `ANSARouter` is depicted in Appendix 8.2. Implementation is fully in compliance with namely [98] and [100], which has been proved in our papers [131], [132] and [133].

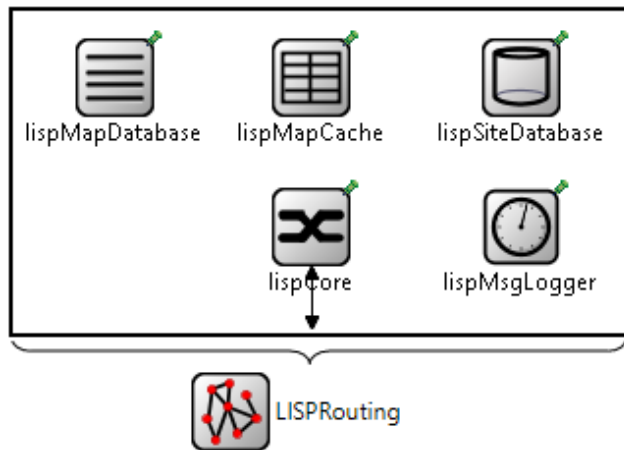


Fig. 38: LISPRouting module structure

All LISP abstract data structures and settings contain dynamic state according to simulation setup and run, or could be statically preconfigured using XML file prior to simulation beginning. Map-cache or map/site database are implemented using generic class `LISPMapStorage` that is extended via C++ inheritance to accommodate different requirements of each control plane component. Every `LISPMapStorage` contains the ordered list of `LISPMapEntry` instances.

Following subchapters contain a brief description of implementation notes regarding each implemented submodule. Illustrative figures refer to the testing scenario depicted in Fig. 39.

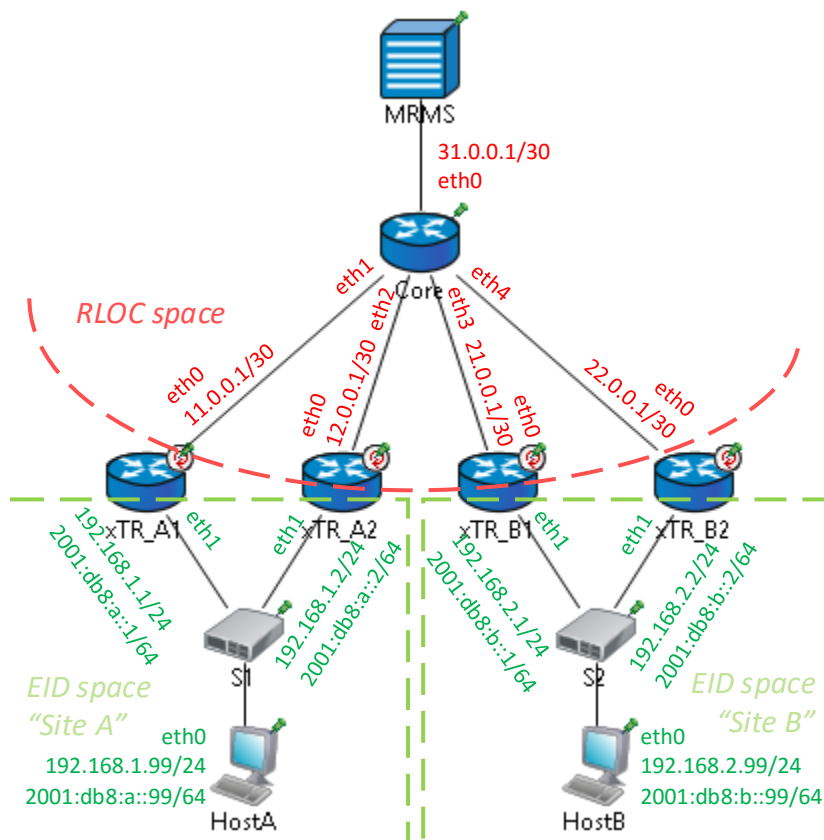


Fig. 39: LISP illustrative scenario

The scenario contains two sites – green areas *Site-A* (interconnected by switch *S1*, bordered by *xTR_A1* and *xTR_A2*) and *Site-B* (interconnected by *S2*, bordered by *xTR_B1* and *xTR_B2*). The network graph contains router *MRMS*, which acts as MR and MS for both sites. IPv4 only capable core (red area) is simulated by a single *Core* router. Static routing is employed to achieve mutual connectivity across the core. *HostA* and *HostB* are dual-stack devices, where *HostA* is scheduled to ping *HostB* after second successful site registration. *MRMS* is allowed to proxy-reply on mapping requests for *Site-A*. All RLOCs are configured with priority 1 and weight 50 to achieve equal load balancing for incoming traffic. This scenario (named “LispHA” located in `/examples/ansa/lispHA`) is contained in contributed LISP source codes thus it is easy to reproduce results.

LISPMsgLogger Submodule

This submodule records and collects statistics about the LISP control plane operation, i.e., number, type, timestamp and size of each sent/received message. The statistics collection is integrated into OMNeT++ as a special signal, and build-in result analysis allows the creation of complex data sets.

LISPCore Submodule

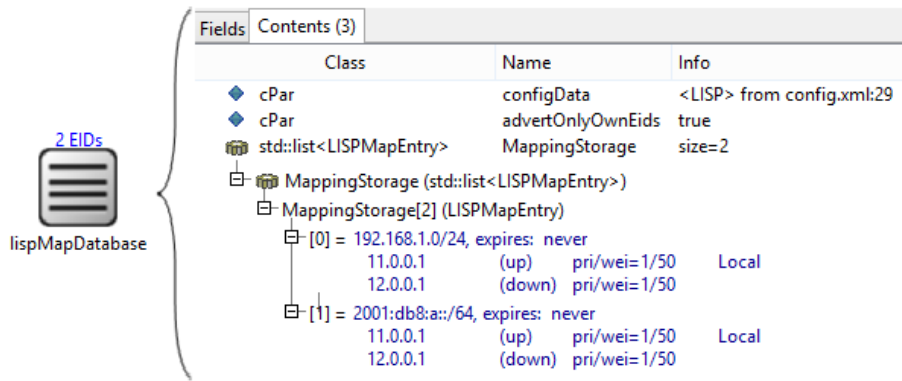
Module independently combines the functionality of ITR, ETR, MR and MS. Each role could be enabled from configuration thus creating different kinds of high-level devices. Roles are connected with following tasks: a) encapsulation and decapsulation of data traffic; b) ETR’s site registrations and MS site maintenance; c) ITR performing mapping lookups; and d) MR delegating queries.

This module handles all LISP control and data traffic. Messages are processed according to finite-state machines that are based on RFC description. Various timer expirations govern some states transitions (e.g., RLOC probing, regular site registration) others by message events (e.g., mapping request-reply scheme). Control messages may cause an internal state change of another LISP submodule such as new mapping added to map-cache or locator state refreshed by RLOC probe. Control messages pass to/from UDP submodule. Data messages are properly encapsulated/decapsulated and passed to appropriate network layer submodule.

LISPMapDatabase Submodule

Each xTR is designated to maintain a state of its LISP sites. This involves responsibility to retrieve results of probed non-local locators or to know, which local interfaces are used for LISP routing. The necessary amount of state information is similar as in the case of Cisco’s control plane for `show {ip|ipv6} lisp database` command [134].

Fig. 40 illustrates map database with two LISP sites ([192.168.1.0/24](#) and [2001:db8:a::/64](#)) and their state.



Fields	Contents (3)		
	Class	Name	Info
◆	cPar	configData	<LISP> from config.xml:29
◆	cPar	advertOnlyOwnEids	true
📦	std::list<LISPMapEntry>	MappingStorage	size=2
📦	MappingStorage (std::list<LISPMapEntry>)		
📦	MappingStorage[2] (LISPMapEntry)		
	[0] = 192.168.1.0/24, expires: never		
	11.0.0.1 (up) pri/wei=1/50		Local
	12.0.0.1 (down) pri/wei=1/50		
	[1] = 2001:db8:a::/64, expires: never		
	11.0.0.1 (up) pri/wei=1/50		Local
	12.0.0.1 (down) pri/wei=1/50		

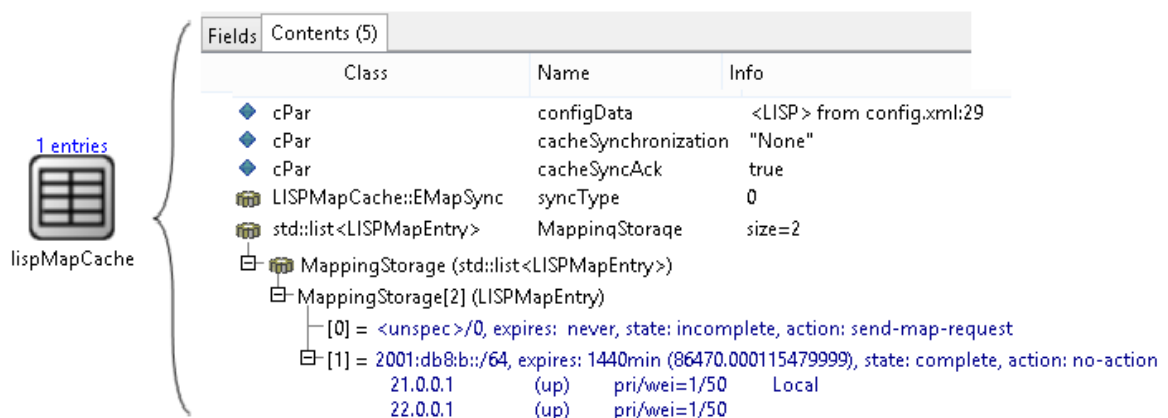
Fig. 40: Content of xTR_A1's LISPMapDatabase

LISPMapCache Submodule

The module contains local map-cache that is populated on demand by routing data traffic between LISP sites. Each record (EID-to-RLOC mapping) has its separate handling (i.e., expiration timer, status, available RLOCs, associated LISP routing action). Submodule also contains synchronization type parameter that tells LISPCore whether to perform map-cache syncing or not. Map-cache lookups are implemented as standard longest prefix match operation that tries to return the most exact EID mapping that is currently available. In real routers, there are separate map-caches for IPv4 and IPv6 EIDs, but in simulator we can afford to simulate them using one abstract data structure. Nevertheless, all entries are sorted according to EID's binary IP address value, where IPv6 come after IPv4. Cisco's control plane maintains similar information, what could be seen in the output of `show {ip|ipv6} lisp map-cache` command [134].

Fig. 41 shows map-cache with two records:

- 0) default one for IPv4 and IPv6 EIDs that are not matched by any subsequent record and that causes LISPCore to initiate mapping query;
- 1) record for EID `2001:db8:b::/64` that is reachable via two locators `21.0.0.1` and `22.0.0.2`.



Fields	Contents (5)		
	Class	Name	Info
◆	cPar	configData	<LISP> from config.xml:29
◆	cPar	cacheSynchronization	"None"
◆	cPar	cacheSyncAck	true
📦	LISPMapCache::EMapSync	syncType	0
📦	std::list<LISPMapEntry>	MappingStorage	size=2
📦	MappingStorage (std::list<LISPMapEntry>)		
📦	MappingStorage[2] (LISPMapEntry)		
	[0] = <unspec>/0, expires: never, state: incomplete, action: send-map-request		
	[1] = 2001:db8:b::/64, expires: 1440min (86470.000115479999), state: complete, action: no-action		
	21.0.0.1 (up) pri/wei=1/50		Local
	22.0.0.1 (up) pri/wei=1/50		

Fig. 41: Content of xTR_A1's LISPMapCache

LISPSiteDatabase Submodule

Submodule contains MS's site database that maintains LISP site registration from ETRs. Each site may have one or more ETR servers, where each one registers set of EIDs. Apart from ETR's independent EID-to-RLOC mappings, LISP site database consist of site-specific parameters such as shared key, proxy capability or registrar's statistics. State information is similar to the content of Cisco's control plane for the `show lisp site detail` command [134].

Illustration in Fig. 42 shows MRMS's site-database with two successfully registered sites: 0) *Site-A* with ETRs 11.0.0.1 and 12.0.0.1; and 1) *Site-B* with ETRs 21.0.0.1 and 22.0.0.1. *Site-A*'s ETRs register EIDs 192.168.1.0/24 and 2001:db8:a::1/64 reachable via RLOCs 11.0.0.1 and 12.0.0.1. *Site-B*'s ETRs register EIDs 192.168.2.0/24 and 2001:db8:b::1/64 reachable via RLOCs 21.0.0.1 and 22.0.0.1.

Fields	Contents (2)	
Class	Name	Info
cPar	configData	<LISP> from config.xml:270
std::list<LISPSite>	SiteDatabase	size=2
SiteDatabase (std::list<LISPSite>)		
SiteDatabase[2] (LISPSite)		
[0] = Site-A, key: "HesloA"		
Maintained EIDs>		
192.168.1.0/24		
2001:db8:a::/64		
Registered ETRs>		
ETR 11.0.0.1, last at: 120.00002378, proxy-reply		
192.168.1.0/24		
11.0.0.1 (up) pri/wei=1/50	Local	
12.0.0.1 (up) pri/wei=1/50	Local	
2001:db8:a::/64		
11.0.0.1 (up) pri/wei=1/50	Local	
12.0.0.1 (up) pri/wei=1/50	Local	
ETR 12.0.0.1, last at: 120.000036579999, proxy-reply		
192.168.1.0/24		
11.0.0.1 (up) pri/wei=1/50	Local	
12.0.0.1 (up) pri/wei=1/50	Local	
2001:db8:a::/64		
11.0.0.1 (up) pri/wei=1/50	Local	
12.0.0.1 (up) pri/wei=1/50	Local	
[1] = Site-B, key: "HesloB"		
Maintained EIDs>		
192.168.2.0/24		
2001:db8:b::/64		
Registered ETRs>		
ETR 21.0.0.1, last at: 140.00002378		
192.168.2.0/24		
21.0.0.1 (up) pri/wei=1/50	Local	
22.0.0.1 (up) pri/wei=1/50	Local	
2001:db8:b::/64		
21.0.0.1 (up) pri/wei=1/50	Local	
22.0.0.1 (up) pri/wei=1/50	Local	
ETR 22.0.0.1, last at: 120.000049379998		
192.168.2.0/24		
21.0.0.1 (up) pri/wei=1/50	Local	
22.0.0.1 (up) pri/wei=1/50	Local	
2001:db8:b::/64		
21.0.0.1 (up) pri/wei=1/50	Local	
22.0.0.1 (up) pri/wei=1/50	Local	

Fig. 42: Content of MRMS's LISPSiteDatabase

4.4.5 Results

This section presents results of evaluation of newly implemented mechanisms. Each measured phenomenon has its subsection with dedicated network graph and scenario. The goal of this subchapter is to show: a) the impact of synchronization on a packet drop rate (and a number of map-cache misses) and to enumerate the burden of deploying it on control plane; and b) the impact of merged RLOC probing on control plane processing.

Impact of Map-Cache Synchronization

We prepared simulation network that contains a LISP site (network EID `192.168.1.0/24` reachable via two RLOCs `11.0.0.1` and `12.0.0.1`) with two routers (*xTR1* and *xTR2*), which provide highly-available VRRP default gateway (`192.168.1.254`) for two hosts interconnected by switch *SW*. *Host1* and *Host2* are pinging IPv4 EIDs (`172.16.[0-19].0/24`) randomly thus generating traffic that triggers LISP mapping system queries. All routing is done statically. Hence, there is no need to employ routing protocol on *Core* router. We prepared special xTR called *xTR_Responder1* that: a) registers destination EIDs to *MRMS*; and b) responds to hosts ICMP messages. The whole network graph is depicted in Fig. 43. Also this scenario (named “LispSyncTest”) is located in `/examples/ansa/lispSyncTest` folder of available source codes.

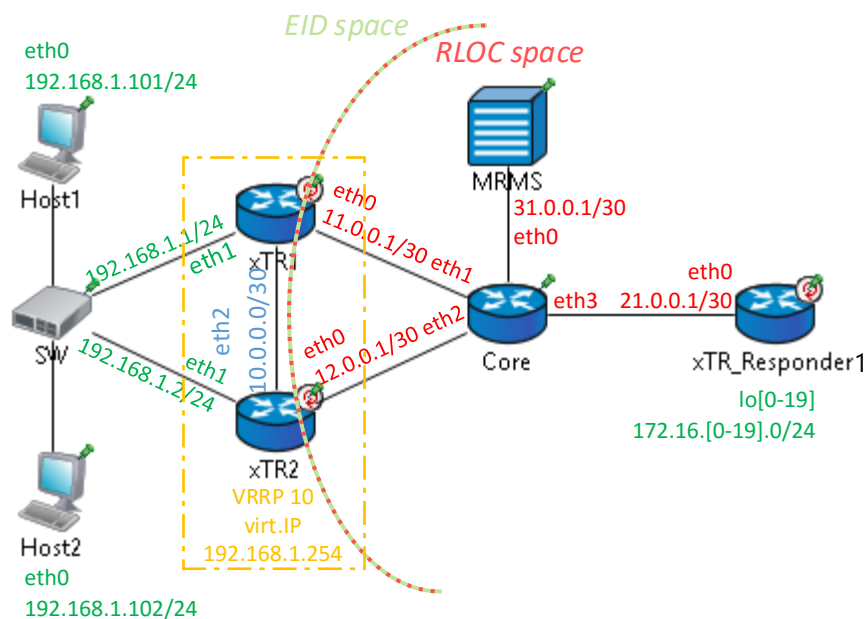


Fig. 43: LISP testing network for Map-Cache synchronization

The testing scenario is focused on cache misses due to the missing mapping rather than expired ones because of default TTL value (1 day). Five minutes time slot with the single VRRP Master outage is the simplest illustration of how to compare the impact of map-cache synchronization. During the outage, all *xTR1*'s interfaces shut down (i.e., they are physically disconnected from the network). The *xTR1*'s control plane is operational (generating scheduled LISP messages, which are not delivered).

We scheduled following phases for the test run focusing on map-cache synchronization:

- #1) At first, all xTR s register their EIDs. In the case of $xTR_Responder1$, EID space is modeled with the help of loopback interfaces – twenty of them ranging with addresses from $172.16.0.0/24$ to $172.16.19.0/24$ reachable via single RLOC $21.0.0.1$. In case of $xTR1$ and $xTR2$, EID $192.168.1.0/24$ is reachable via two RLOCs $11.0.0.1$ and $12.0.0.1$;
- #2) $xTR1$ and $xTR2$ form VRRP group with VID 10 and virtual address $192.168.1.254$, which is used by $Host1$ and $Host2$ as default-gateway. $xTR1$ is Master because of higher priority ($xTR1$ has 150, $xTR2$ only 100) as long as it is operational.
- #3) $Host1$ starts pinging ten random EIDs in the range from $172.16.0.0/24$ to $172.16.9.0/24$. Because EIDs are chosen randomly, they may be duplicate. Each first ICMP packet causes mapping query and is dropped.
- #4) Then right before a new LISP registration (at $t=119s$), $xTR1$ failure occurs. Hosts traffic is diverted to a new VRRP Master, which is $xTR2$.
- #5) After phase 4), also $Host2$ starts to ping ten random EIDs from $172.16.10.0/24$ to $172.16.19.0/24$. Same duplicity rule as in 3) applies.
- #6) $xTR1$ recovers from the outage at $t=235s$ and once again all hosts traffic goes through it.

Depending on the map-cache synchronization type, additional map-cache misses might occur. $xTR1$ and $xTR2$ synchronized themselves via their RLOCs ($11.0.0.1$ for $xTR1$ and $12.0.0.1$ for $xTR2$).

The scenario has been tested with three simulation configurations, which we can divide according to the used map-cache synchronization technique: α) no synchronization at all (default LISP behavior); β) *naïve* mode; and γ) *smart* mode. Impact on map-cache is summarized in Tab. 9 for all previously mentioned different configuration runs. Fewer map-cache misses are considered better.

We do not employ LISP synchronization acknowledgment scheme for β/γ -runs, the impact of acks is analyzed later. The scenario offers testing of all three kinds of addressed for SS member identification – e.g., nonLISP with $10.0.0.0/30$; RLOC with $11.0.0.1$ and $12.0.0.1$; and EID with $192.168.1.1$ and $192.168.1.2$) with same results. Nevertheless, we use EIDs as the most feasible options.

Before interpreting results, please note that $Host1$ randomly (using same random generator seeds) chose eight different EIDs, $Host2$ six EIDs, fourteen various ping destinations in the summary.

Phase	α cache misses		β cache misses		γ cache misses	
	$xTR1$	$xTR2$	$xTR1$	$xTR2$	$xTR1$	$xTR2$
#3	8	0	8	0	8	0
#5	0	14	0	6	0	6
#6	14	0	0	0	0	0
Total	22	14	8	6	8	6

Tab. 9: Count of map-cache misses under different configurations in scenario with one outage

Without any synchronization, traffic diversion to a new VRRP Master always causes misses due to unknown mappings. We can see it in phases #5 and #6 for α -run when the router starts to dispatch LISP data with the empty map-cache.

If synchronization is employed, then, only new destinations lead to map-cache miss. This is because a new VRRP Master already has mappings discovered by neighbor xTR. Hence, there is a difference in phase #5 for α -run (empty cache) and β/γ -runs (cache in sync with SS member). The difference (36 cache misses versus 14) would be even more significant in the case of multiple VRRP Master outages. Please note that every map-cache miss is also connected with the data packet drop.

In order to compare synchronization modes, we conducted measurement taking into account all LISP control messages processed by LISPCore module, namely their packet sizes. We assume that larger size is always a greater burden for router's control plane processing. Fig. 44 shows results (α -run = blue crosses, β -run = green triangles, γ -run = red circles), where each symbol represents one LISP control message.

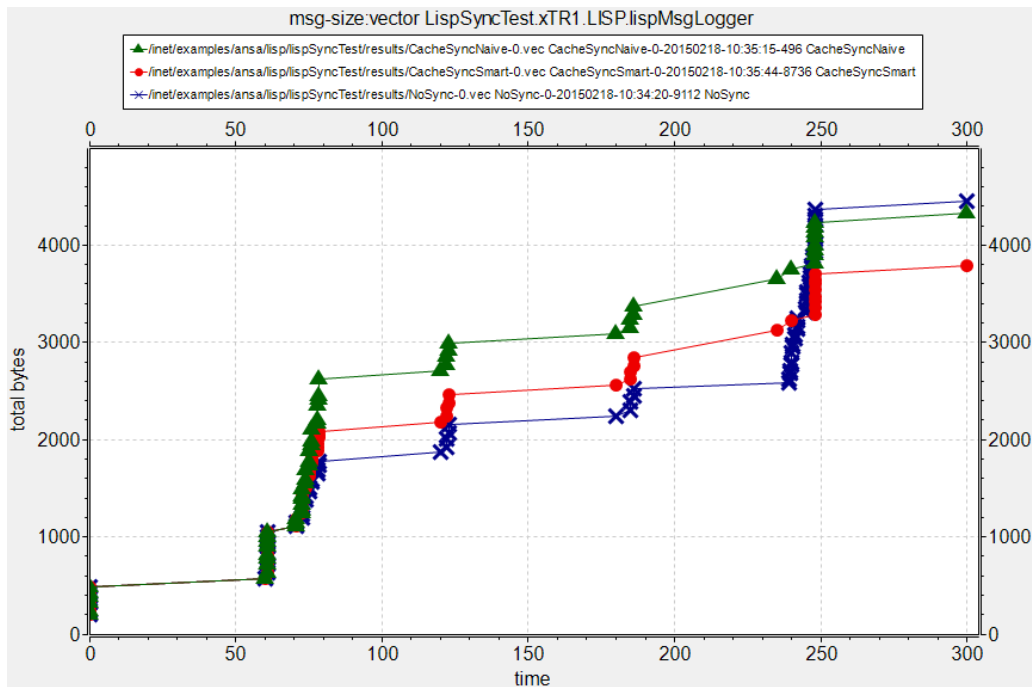


Fig. 44: xTR1's LISP control messages occurrence and total processed byte size in scenario with single outage

We can see that *smart* outperforms *naïve* because it is less intensive while only single mapping is transferred during synchronization, not a whole map-cache. Moreover, both synchronization modes are better than no synchronization on protocol overhead because they decrease the number of mapping queries (i.e., exchanged messages count). The difference is not so significant on Fig. 44, especially between *naïve* and no sync mode. However, it is getting more obvious as the number of VRRP outages increases. Following table and figure prove this claim for the same network but with two xTR1 outages – basically phases #4 and #6 repeat twice.

Phase	α cache misses		β cache misses		γ cache misses	
	$xTR1$	$xTR2$	$xTR1$	$xTR2$	$xTR1$	$xTR2$
#3a	8	0	8	0	8	0
#5a	0	14	0	6	0	6
#6a	14	0	0	0	0	0
#5b	0	0	0	0	0	0
#6b	14	0	0	0	0	0
Total	36	14	8	6	8	6

Tab. 10: Count of map-cache misses under different configurations in scenario with two outages

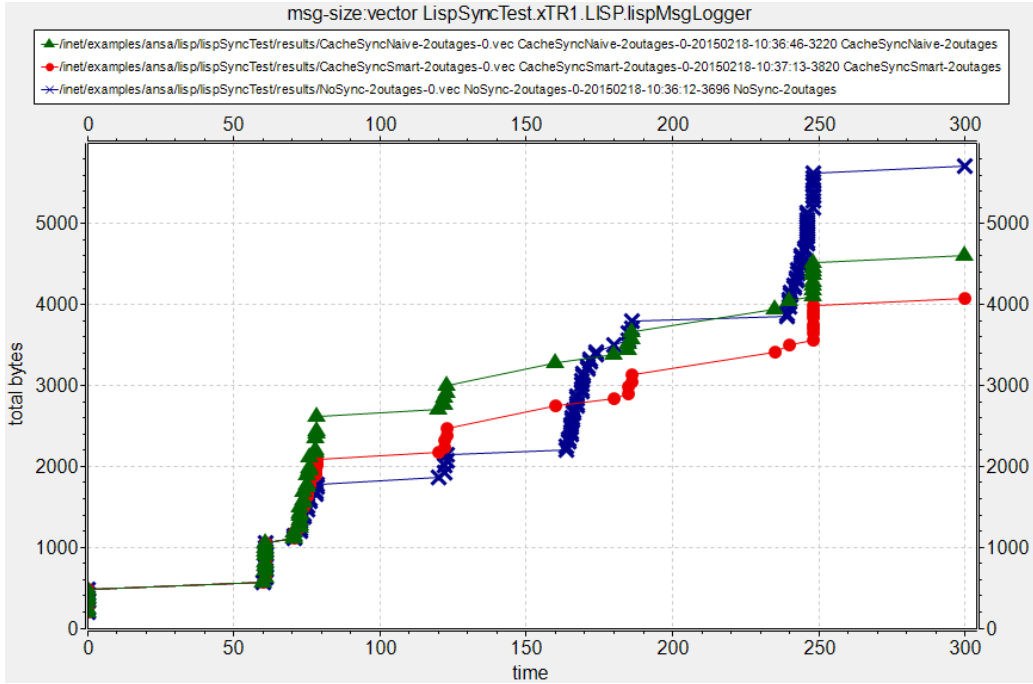


Fig. 45: $xTR1$'s LISP control messages occurrence and total processed byte size in scenario with two outages

Repetition of phases 4), 5) and 6) is denoted in Tab. 10 with letters: “a” for the first outage; and “b” for the second outage. In Tab. 10, we can observe that a total number of cache misses for α -run has increased by 14. $xTR1$ had gone down (losing its map-cache content), then went back (repopulating map-cache once again with 14 EIDs) and then this cycle repeats once again. For β -run and γ -run, additional outages pose no change, because $xTR1$ completely synchronizes itself with $xTR2$ ($xTR2$ sends the whole map-cache as soon as it detects the status of the one of $xTR1$'s RLOCs up), when it is once again operational. Fig. 45 shows an increase in a number of processed LISP control message for no synchronization, where impacts of other synchronization techniques remain same.

LISP synchronization acknowledgment mechanism poses an additional control plane burden. In order to evaluate acknowledgment impact, we conducted measurement on the same network with two outages. The results in a number of processed LISP control messages bytes are depicted in Fig. 46 and can be compared with Fig. 44.

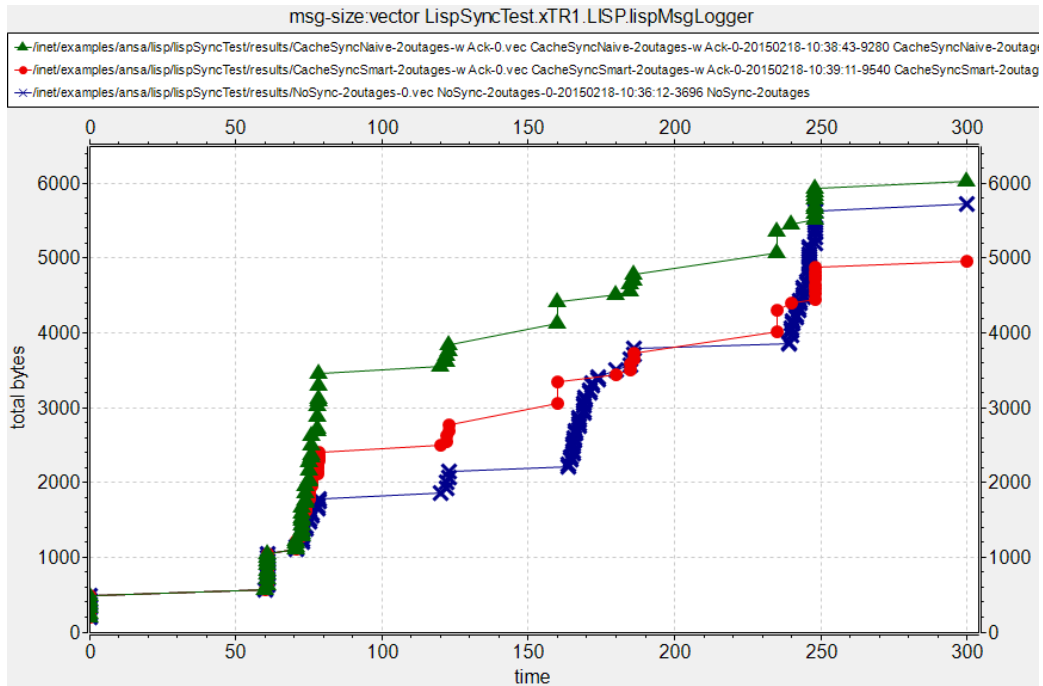


Fig. 46: xTR1’s LISP control messages occurrence and total processed byte size in scenario with two outages + ack

It is apparent that protocol overhead on the number of messages has increased. In the case of no synchronization, it slightly outperforms *naïve* mode by a total size of processed bytes. However, the *smart* mode still has the best characteristic even with enabled acknowledgments. Once again, we can expect that additional outages or more EID ping destinations would influence results in favor of β/γ -runs over α -run.

To summarize the evaluation of map-cache synchronization technique, we provide Tab. 11, which shows $\alpha/\beta/\gamma$ -run (i.e., none, *naïve* and *smart* sync) statistics for different scenarios (i.e., one/two/three outage(s) with or without acknowledgment). xTR1’s statistic numbers are depicted with following column meanings: “miss” as the number of map-cache miss occurrence; “cnt” as the total count of LISP control plane messages sent and received; “size” as processed messages count by LISP control plane measured in total byte size. We added to Tab. 11 also same statistics section for the scenario with three outages in order to analyze trends even thou that it is not described via dedicated table and graph above (nevertheless, we appended them to Addendum 8.3.2 for completeness). Results show a linear growth in complexity.

single xTR1 outage scenario									single xTR1 outage with sync ack scenario											
α			β			γ			α			β			γ					
miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size
22	81	4 458	8	62	4 328	8	62	3 796	22	81	4 458	8	71	5 458	8	71	4 394			
two xTR1 outages scenario									two xTR1 outages with sync ack scenario											
α			β			γ			α			β			γ					
miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size
36	109	5 718	8	63	4 614	8	63	4 082	36	109	5 718	8	73	6 030	8	73	4 966			
three xTR1 outages scenario									three xTR1 outages with sync ack scenario											
α			β			γ			α			β			γ					
miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size	miss	cnt	size
50	137	6 978	8	64	4 900	8	64	4 368	50	137	6 978	8	75	6 602	8	75	5 538			

Tab. 11: xTR1’s statistics for different map-cache synchronization scenarios

Impact of Merged RLOC Probing

We took the previous network and adjusted it. Currently, it contains a LISP site with just one *xTR* router and one end-device called *Host1*. More important are LISP sites that are reachable via *xTR_Responder1* and *xTR_Responder2*. We simulate multiple EID networks reachable via the same xTRs with the help of loopback interfaces. Each *xTR_Responder* has forty loopbacks with EID addresses in the range of `172.16.[0-39].0/24`. Each EID is being registered towards *MRMS* as reachable via *xTR_Responder1*’s RLOC `21.0.0.1` and *xTR_Responder2*’s RLOC `22.0.0.1`. VRRP functionality on *xTR* is disabled because it is not needed for this scenario. *Host1* might randomly generate ICMP traffic towards destination EIDs, but this is not necessary for merged RLOC probing analysis. All communicating parties are interconnected via *Core* employing static routing configuration. The whole network graph is depicted in Fig. 47. Also this scenario (named “LispProbeTest”) is located in `/examples/ansa/lispProbeTest` folder of available source codes.

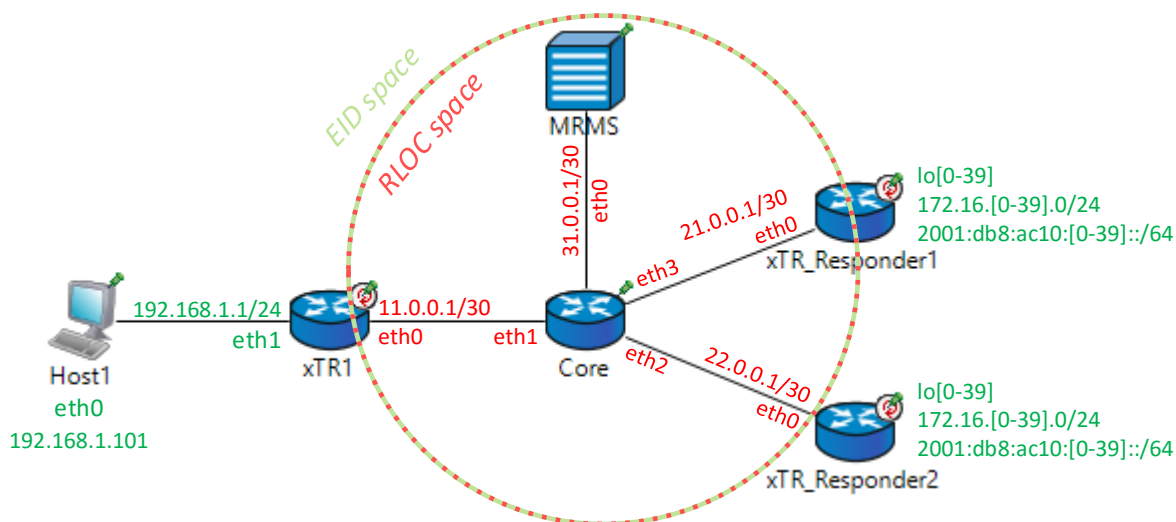


Fig. 47: LISP testing network for merged RLOC probing

RLOC probing starts immediately after LISP routing control plane is initialized. Following phases occur no matter on used RLOC probing algorithm:

- Probing xTR sends *LISP Map-Request Probe* to RLOC address for a given set of EIDs;
- Probed xTR responds with *LISP Map-Reply Probe* announcing that RLOC is up;
- In case that *LISP Map-Request Probe* was not replied, probing xTR repeats the probe at time $t_{next} = t_{last} + 2^{numOfRetries}$, where t_{last} is the time last probe was sent and $numOfRetries$ is a number of retry attempts to send this probe. By default, after three unsuccessful *LISP Map-Request Probe*, RLOC is marked as down and the next probe is scheduled after 60 seconds.

Optional phase 3) behavior is solely based on Cisco implementation observations. Also Cisco’s LISP implementation has some other specifics: a) postponed start of first EID registration ($t + 60$ seconds since control plane initialization); b) postponed start of RLOC probing for IPv6 RLOCs ($t + 30$ since the first IPv4 probe). We have integrated this behavior into the LISP simulator. However, we are not employing it in order to provide better readability of this scenario’s results.

These phases repeat by default every minute to keep RLOC reachability up-to-date. This interval could be decremented to a lower value, but protocol overhead increases in an inverse relationship.

Measurement is focused on a number of *LISP Map-Request/Reply Probes* exchanged between *xTR_Responder1* and *xTR_Responder2* and the amount of corresponding bytes processed by *xTR_Responder1*’s LISP control plane. We assume that five minutes simulation time is a period long enough to show the trend of each RLOC probing algorithm. During this time, five RLOC probe batches occur. Except mandatory EID registrations, no other LISP control traffic is spoiling the results.

We have conducted two simulation scenarios in order to observe complexity trends. The first one is for the network with forty different EIDs (twenty IPv4 [172.16.\[0-19\].0/24](#) and twenty IPv6 [2001:db8:ac10:\[0-19\]::/64](#)) on xTR_Responders reachable via RLOCs [21.0.0.1](#) and [22.0.0.1](#), the second with eighty different EIDs (forty IPv4 [172.16.\[0-39\].0/24](#) and forty IPv6 [2001:db8:ac10:\[0-39\]::/64](#)). All three algorithms are evaluated separately as different configuration simulation runs - Cisco’s default algorithm as δ -run, *simple* as ϵ -run and *sophisticated* as λ -run algorithm variants of merged RLOC probing.

40 EIDs scenario						80 EIDs scenario					
δ		ϵ		λ		δ		ϵ		λ	
cnt	size	cnt	size	cnt	size	cnt	size	cnt	size	cnt	size
805	55 500	25	8 520	25	28 530	1 605	110 900	25	15 920	25	56 330

Tab. 12: xTR_Responder1’s statistics for different RLOC probing algorithm scenarios

Total count of sent and received LISP control messages are shown in Tab. 12. Columns have following meaning: “cnt” as the total count of LISP control plane messages sent and received; “size” as the amount processed messages by LISP control plane measured in total byte size.

Apart from five *LISP Map-Register*, *xTR_Responder1* five times: a) sends *LISP Map-Request Probe* and receives *LISP Map-Reply Probe*; b) receives *xTR_Responder2*'s probes and responds to them with replies. It is apparent that a count of exchanged messages is drastically lower when using any merged RLOC probing algorithm. Cisco's algorithm generates RLOC probe for each EID-to-RLOC mapping, which means forty/eighty *LISP Map-Request Probe* and forty/eighty *LISP Map-Reply Probe* messages per single phases #1 and #2 occurrences. Opposite to that any merged RLOC algorithm exchanges only single *LISP Map-Request/Reply Probe* pair between *xTR_Responders*.

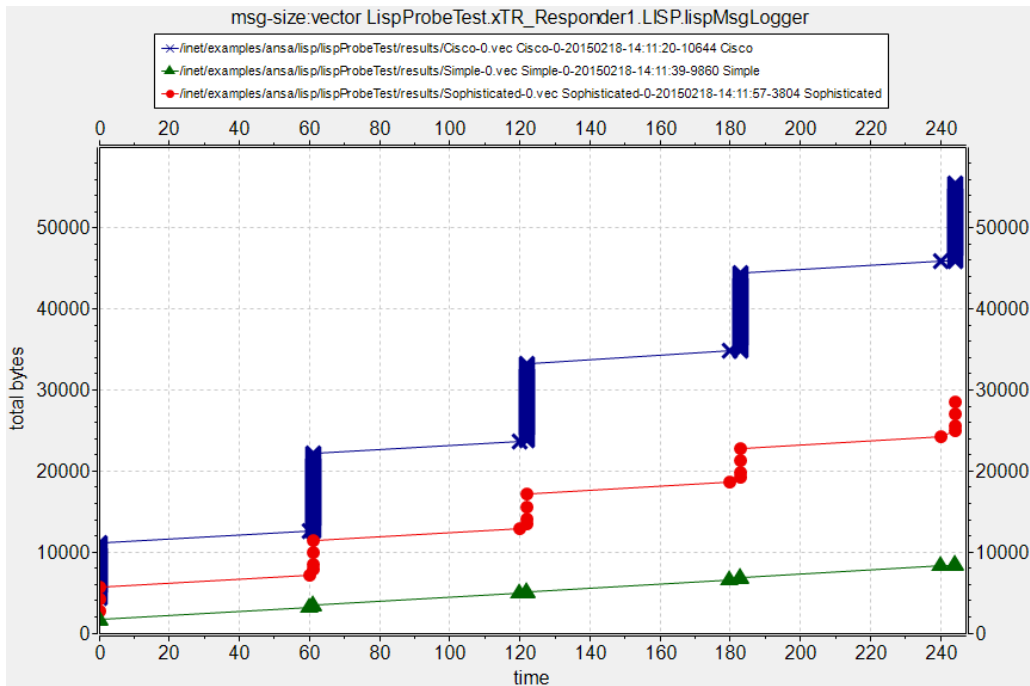


Fig. 48: *xTR_Responder1*'s LISP messages occurrence and total processed byte size in scenario with forty EIDs

In Fig. 48, we can see that ϵ -run has better protocol overhead measured in the total amount of bytes processed by *xTR_Responder1*. This is because each probe carries only single EID chosen in a round-robin fashion, where successful reception of *LISP Map-Reply Probe* refreshes RLOC state for all EIDs that are using it. In the case of the *sophisticated* algorithm, all relevant EIDs are packed in single probe thus (significantly) increasing its size (but still half of Cisco's total processed byte size). On the other hand *simple* merged RLOC probing algorithm might seem to be too simple and lacking of accuracy if we want the use-case where the same RLOC is up for some EIDs, and down for another EIDs. In that case, *sophisticated* variant offers the same functionality but with better granularity.

For completeness, the same graph as Fig. 48 but for a scenario with eighty EIDs is in Addendum 8.3.3. Because scenarios are linearly dependent, the only difference is in Y-axis values and a higher amount of RLOC probe (symbol) occurrences.

4.5 Chapter Summary

This chapter described in great detail all routing aspects of LISP. In the first subchapter, we started with a basic overview of LISP functionality and its main components. We focused on the distributed mapping system of LISP including how map-cache content impacts LISP routing performance. We outlined LISP signalization messages together with their syntax and semantics. We discussed ways how LISP coexists with traditional TCP/IP networks and what are transition possibilities and deployment options.

In the next subchapter, we demonstrated LISP theory of operation on concrete examples. We started with simple unicast data transfer focusing on map-and-encap parts of packet handling. Proceeding next, we illustrated mapping system behavior including LISP site registration, EID-to-RLOC mapping query, and subsequent response. As the last demonstration, we depicted communication between LISP and the non-LISP world.

The Subchapter 4.3 provided Rather a brief information regarding LISP global deployment (i.e., LISP BetaNetwork) and available vendor's implementations. We also mentioned existing simulators and simulation modules that have been used for LISP research in the past.

The last subchapter described one of the main contributions of this thesis. Two major issues are introduced that limit LISP operation – Site-based Synchronization Problem and Locator Path Liveness Problem. Furthermore, we proposed specific map-cache synchronization techniques and merger RLOC probing algorithms, which should reduce protocol overhead and increase LISP routing performance. Hence, these improvements should at least partially deal with problems above. Moreover, we developed and implemented brand new simulation modules of LISP (and as a byproduct also VRRP) intended for OMNeT++. Employing these modules, we tested and successfully proved the effectiveness of proposed improvements.

If we want to qualify and quantify impact of our propositions the following items hold:

- Both *naïve* and *smart* map-cache synchronization modes significantly reduce (theoretically to zero) map-cache misses for sites with multiple ITRs;
- *Smart* mode outperforms *naïve* mode in protocol overhead (in number of processed bytes):
 - having approx. 11% lower overhead for scenarios without acknowledgment and both are better than no synchronization;
 - having approx. 17% lower overhead for scenarios with acknowledgment, where *smart* is always better than no synchronization, and *naïve* gets better with more outages;
- Merged RLOC probing decreases radically protocol overhead (in processed bytes count) of locator liveness checking:
 - the *simple* algorithm reduces overhead by approx. 85%;
 - the *sophisticated* algorithm reduces overhead by approx. 50%;

5 Recursive Internet Architecture

- ☞ *–“In order to understand recursion, one must first understand recursion.” Anonymous*
- ☞ *What is RINA and what are its most distinctive features?*
- ☞ *What is technology readiness level of RINA?*
- ☞ *Can we prove RINA’s feasibility as the clean-slate architecture?*

RINA is the clean-slate architecture aimed to change the whole Internet unlike just temporary fixes for current status quo. RINA concept is based on John Day’s thoughts, lectures and book [3] regarding ISO/OSI initiative failure, TCP/IP development, commercial adoption of the Internet and other technical/political events in Internet history (see Addendum 8.4 for more).

While proposals and approaches discussed in the previous chapters deal with extending or correcting the current Internet architecture, RINA attempts to define a novel Internet architecture, RINA is a continuation of the original internetworking ideas from the mid-1970s. The architecture as proposed by RINA is fundamentally different from the current TCP/IP networking. The RINA approach is based on a few principles instead of a broad and complex eco-system of modern Internet. The idea of the recursive composition of layers arises naturally from the structure of repeating computer networking patterns. Instead of strictly separating network functions into a predefined set of layers, RINA enables to compose a stack from layers that may offer a nearly the same set of functions. In RINA, each layer only has to provide data transfer between nodes of the layer. Depending on the other functionality represented as mechanisms and policies, the RINA nodes can communicate reliably or securely. Another difference to the current TCP/IP’s Internet is that every communication in RINA is considered as communication between a pair of networking processes regardless the layer at which this communication occurs. Assuming the single communication paradigm simplifies the overall design. It was shown that only a couple primitive operations need to be implemented in the communication protocol. Also, all layers employ the same protocols which contrast to TCP/IP model in which each layer defines its set of protocols. RINA was designed to provide a simpler and efficient alternative to the current Internet architecture.

This chapter familiarizes the reader with RINA basics. Based on our experience, we must admit that “mental-shift” from nowadays networking towards RINA is not easy at all. Hence, the reader is advised to seek further in related references when confused.

Among main goals of this chapter are the following items: a) to introduce RINA as a new networking paradigm; b) to provide in-depth explanation of RINA’s operation; c) to revisit and improve some of RINA specifications; d) to develop the first RINA simulator as a new educational and research tool; and e) to demonstrate RINA theory on practical example employing our enhancements of enrollment and flow (de)allocation procedures.

5.1 Overview

This subchapter introduces theoretical background. However, explanation of the whole Recursive Internet Architecture is far beyond the scope of this thesis. Hence, only parts relevant to the current RINASim functionality are captured. Synthesis of RINA information provided below comes from the following sources: [135], [136], [137], [138] and [139].

5.1.1 Nature of Applications and Application Protocols

Is *application* a part of IPC environment or not? The set of Internet applications was rather simplistic before WWW – one application with a single instance using only one protocol. Hence, there is nearly no distinction between an application and its networking part. However, the web completely changed this situation – one application protocol may be used by more than one application and also one application may have many application protocols.

Following terms are recognized in the frame of RINA, and their relationship is depicted in Fig. 49:

- **Application Process (AP)** – Program instantiation to accomplish some purpose;
- **Application Entity (AE)** – AE is the part of AP, which represents application protocol and application aspects concerned with communication.

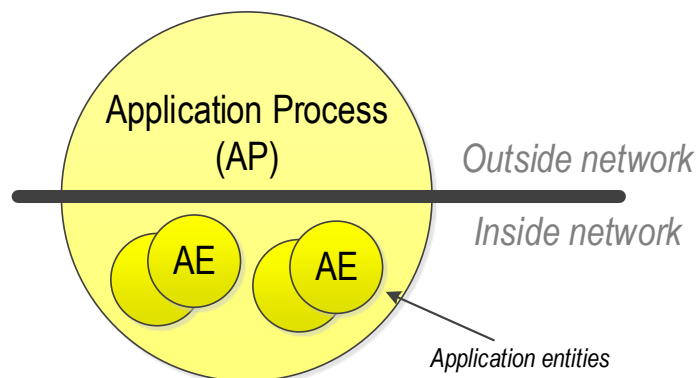


Fig. 49: Application Protocol and Application Entities relationship

There may be multiple instances of the Application Process in the same system. AP may have multiple AEs, each one may process different application protocol. There also may be more than one instance of each AE type within a single AP.

All application protocols are stateless; the state is and should be maintained in the application. Thus, all **application protocols** modify shared state external to the protocol itself on various objects (e.g. data, file, HW peripherals). Because of that, there is only one application protocol that contains atomic operations (e.g., read/write, start/stop). **Data transfer protocols** modify state internal to the protocol, the only external effect is the delivery of SDUs.

5.1.2 Core Terms

The data transport and internetworking tasks together (generally known as *networking*) are functions of **inter-process communication (IPC)**. IPC between two APs on the same operating system needs to locate processes, evaluate permission, pass data, schedule tasks and manage memory. IPC between two APs on different systems works similarly plus adding functionality to overcome the lack of shared memory.

In traditional networking stack, the layer provides a service to the layer immediately above it. As RINA name suggests, recursion and repeating of patterns are the main feature of the robust architecture. Layer recursion became more popular even in TCP/IP with technologies like Virtual Private Networks (VPNs) or overlay networks (e.g., OTV⁶⁹). Recursion is a natural thing whenever we need to affect the scope of communicating parties. However, so far it was just recursion of repeating functions in existing layers. RINA is based on following core ideas:

— “*Networking is interprocess communication...and IPC only!*” [140]

— “*Application Processes communicate via a service provided by a distributed application that provides IPC. The application processes that make up this Distributed IPC Facility provide a protocol that implements an IPC mechanism, and a protocol for managing distributed IPC (routing, security and other management tasks).*” [141]

In ISO/OSI or TCP/IP, there is a set of layers each with entirely different functions. RINA, on the other hand, yields idea of the single generic layer with fixed mechanisms but configurable policies. This layer is in RINA called **Distributed IPC Facility (DIF)** – a set of cooperating APs providing IPC. There is not a fixed number of DIFs in RINA; we can stack them according to application or network needs. From the DIF point of view actual stack depth is irrelevant, DIF may provide a service to (N+1)-layer above and use the service of the (N-1)-layer below. DIF stacking partitions network into smaller, thus, more manageable parts.

The concept of RINA layer could be further generalized to **Distributed Application Facility (DAF)** – a set of cooperating APs in one or more computing systems, which exchange information using IPC and maintain shared state. A DIF is a DAF that does only IPC. **Distributed Application Process (DAP)** is a member of a DAF. **IPC Process (IPCP)** is an AP within DIF delivering inter-process communication. IPCP is an instantiation of DIF membership; computing system is container for IPCPs that perform IPC with other DIF members. An IPCP is specialized DAP. The relationship between all newly defined terms is depicted in Fig. 50.

⁶⁹ Overlay Transport Virtualization (OTV). For more, see <http://www.cisco.com/c/en/us/solutions/data-center-virtualization/overlay-transport-virtualization-otv/index.html>

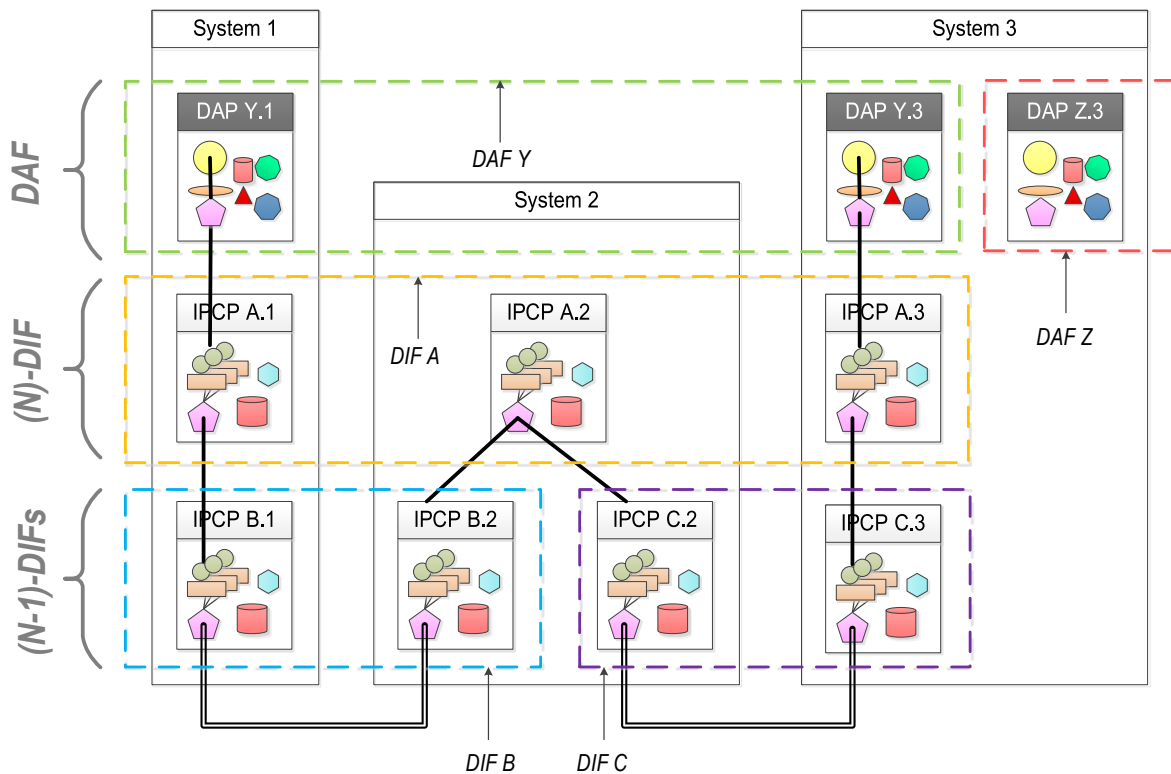


Fig. 50: DIF, DAF, DAP and IPCP illustration

DIF limits and encloses cooperating processes in the one scope. However, its functionality is more general and versatile apart from rigid TCP/IP layers with dedicated functionality (i.e., data-link layer for adjacent node communication, a transport layer for reliable data transfer between applications). DIF provides IPC to either another DIF or to DAF. Therefore, DIF uses a single application protocol with generic primitive operations to support intra-DIF communication.

5.1.3 Connection-oriented vs. Connectionless

The clash between connection-oriented and connectionless approaches (that also corrupted ISO/OSI tendencies) is from RINA perspective quite easy to settle. Connection-oriented and connectionless communication are both just functions of the layer that should not be visible to applications. Both approaches are equal, and it depends on how application requirements are going to be imposed by layer (i.e., which approach is going to be used). On the other hand, connection-oriented limits the dissemination and tends toward static resource allocation. The first one is good for low volume stochastic traffic. The second one is useful for scenarios with deterministic traffic flows.

If the applications request the allocation of communication resources, then layer determines what mechanisms and policies to use. Allocation is accompanied by access rights and description of QoS demands (e.g., what minimum bandwidth or delay is needed for correct operation of application).

5.1.4 Delta-t Synchronization

All properly designed data transfer protocols are soft-state. There is no need for explicit state synchronization (hard-state) and tools like SYNs and FINs are unnecessary.

Initial synchronization of communicating parties is done with the help of **Delta-t** protocol (see [5] and [142]). Delta-t was developed by Richard Watson as the proof-of-concept that time-based synchronization technique is necessary and sufficient for reliable data transfer. He proved that conditions for distributed synchronization were met if the next three timers are realized: a) **Maximum Packet Lifetime (MPL)**, which denotes the upper bound time (value MPL) that a packet can exist in a network; b) Retransmission-timer specifies maximum period (value R) that a sender is willing to retransmit its unacknowledged messages; c) Acknowledgment-timer defines maximum delay (value A) that the receiver of data can wait before sending acknowledgment. Delta-t's main variable Δt is enumerated as:

$$\Delta t = MPL + R + A$$

Delta-t assumes that all connections exist all the time. Synchronization state (e.g., sequence numbers) is maintained only during the active data transfer, but after maximum $2\Delta t$ (on receiver's side) or $3\Delta t$ (on sender's side) periods without any traffic state may be discarded which effectively resets the connection. Because of that, there are no hard-state (with explicit synchronization) protocols only soft-state ones. Delta-t postulates that port allocation and synchronization are distinct.

5.1.5 Separation of Mechanism and Policy

We understand terms *mechanism* and *policy* as they have been defined in Subchapter 2.1. Just to remind the reader that mechanism is invariant, the policy is variant part of any IPC. In the same subchapter, the most common mechanisms has been cataloged using ontology. Nevertheless, this mechanism list is not final and even to several mechanisms exist dozens/hundreds of different policies, how exactly are these mechanisms implemented and enforced.

If we focus only on mechanisms connected with data transfer, then we can clearly separate them into two groups:

- **tightly-bound** that must be associated with every PDU, which handle fundamental aspects of data transfers (e.g., the sequence number of every PDU, integrity check using hashes associated with the PDU content);
- **loosely-bound** that could be associated with data transfer PDUs (but there is no requirement that these mechanisms must be associated with them), which provide additional features (namely reliability and flow control).

Both groups are coupled through **state vector** maintained separately per flow; every active flow has its state vector holding state information. Tightly-bound mechanisms (e.g., ordering of sequence

number) write to state vector, whereabouts loosely-bound mechanisms (e.g., loss detection) read it. For instance, the behavior of retransmission and flow control can be heavily influenced by chosen policies and they can be used independently on each other.

This and the use of abstract/concrete syntax implies that only single generic data transfer protocol based on Delta-t is needed, which may be governed by different transfer control policies. This data transfer protocol modifies state internal to its PM, where application protocol (carried inside) modifies state external to PM.

5.1.6 Naming and Addressing

Application Process communicates in order to share state. In 5.1.1, we mentioned that AP consists of AEs. We need to differentiate between different APs and also different AEs within the same AP. Thus, RINA is using **Application Process Name (APN)** as globally unambiguous, location-independent, system-dependent name. **Application Process Instance Identifier (API-id)** differentiates between multiple instances of the same AP in the system. **Application Entity Instance Identifier (AEI-id)**, which is unambiguous for a single AP, helps us to identify different AE instances of same **Application Entity Name (AEN)** within AP. **Application Naming Information (ANI)** references a complete set of identifiers to name particular application; it consists of four-tuple APN, API-id, AEN, and AEI-id. The only required part of ANI is APN; others are optional. **Distributed Application Name (DAN)** is globally unambiguous name for a set of system-independent APs.

IPC Process has APN to identify it among other DIF members. A RINA **address** is a synonym for IPCP's APN with a scope limited to the layer and structured to facilitate forwarding. APN is useful for management purposes but not for forwarding. Address structure may be *topologically dependent* (indicating the nearness of IPCPs). APN and address are simply two different means to locate an object in different contexts. There are two local identifiers necessary for IPCP functionality – port-id and connection-endpoint-id. **Port-id** binds this (N)-IPCP and (N+1)-IPCP/AP; both of them use the same port-id when passing messages. Port-id is returned as a handle to the communication allocator and is unambiguous within a computing system. **Connection-endpoint-id (CEP-id)** identifies a shared state of one communication endpoint. Since there may be more than one flow between the same IPCP pair, it is necessary to distinguish them. For this purpose, **Connection-id** is formed by combining source and destination CEP-ids with QoS requirements descriptor. CEP-id is unambiguous within IPCP and Connection-id is unambiguous between a given pair of IPCPs. Fig. 51 depicts all relevant identifiers between two IPCPs.

Watson's delta-t implies port-id and CEP-id in order to help separate port allocation and synchronization. RINA's **connection** is a shared state between N-PMs – ends identified by CEP-ids. RINA's **flow** is when connection ends are bound to ports identified by port-ids. The lifetimes of flow and its connection(s) are independent of each other.

The relationship between node and PoA is relative – node address is (N)-address, and its PoA is (N-1)-address. Routes are sequences of (N)-addresses, where (N)-layer routes based on this addresses (not according to (N-1)-addresses). Hence, the layer itself should assign addresses because it understands address structure.

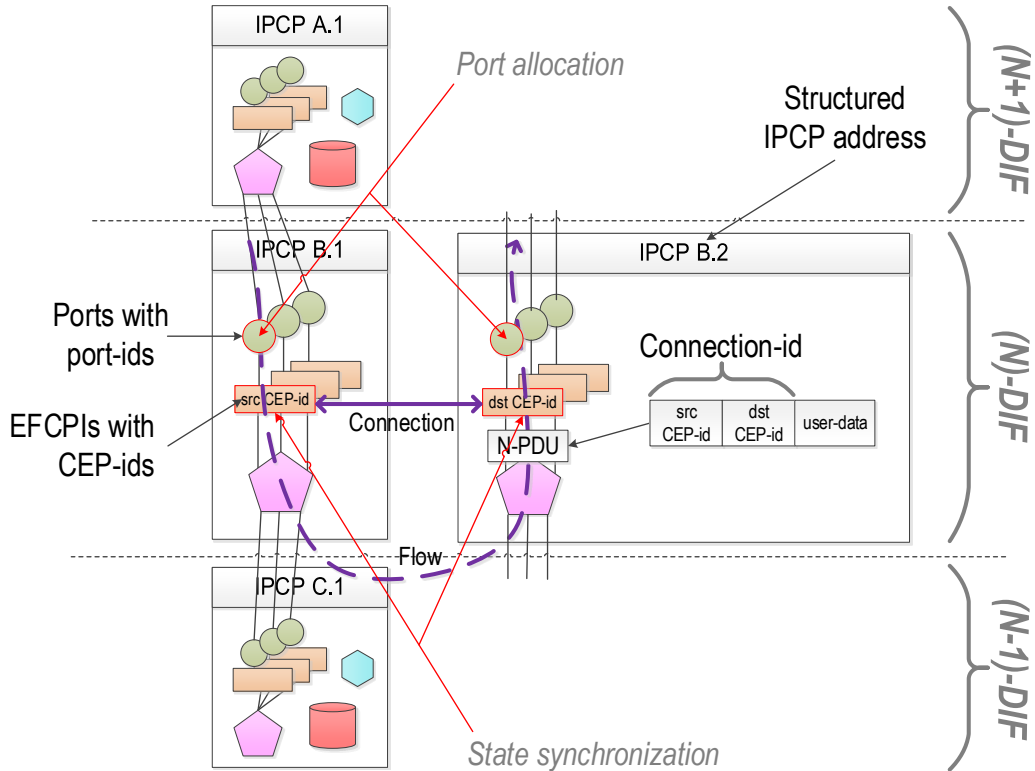


Fig. 51: IPCP local identifiers overview

5.2 RINA Components

To understand RINA architecture means to understand each of its elements. This subchapter starts with a description of high-level RINA network nodes and then goes deeper and outlines various IPC Management and IPCP components.

5.2.1 Nodes

There are only three basic kinds of nodes in RINA network (illustrated in Fig. 52). Each kind represents computing system running RINA:

- **Hosts** – end-devices for IPC containing AEs in the top layer; they employ two or more DIF levels;
- **Interior routers** – interim devices, which are interconnecting (N)-DIF neighbors via multiple (N-1)-DIFs; they employ two or more DIF levels;
- **Border routers** – interim devices, which are interconnecting (N)-DIF neighbors via (N-1)-DIFs, where some of (N-1)-DIFs are reachable only through (N-2)-DIFs; they employ three or more DIF levels.

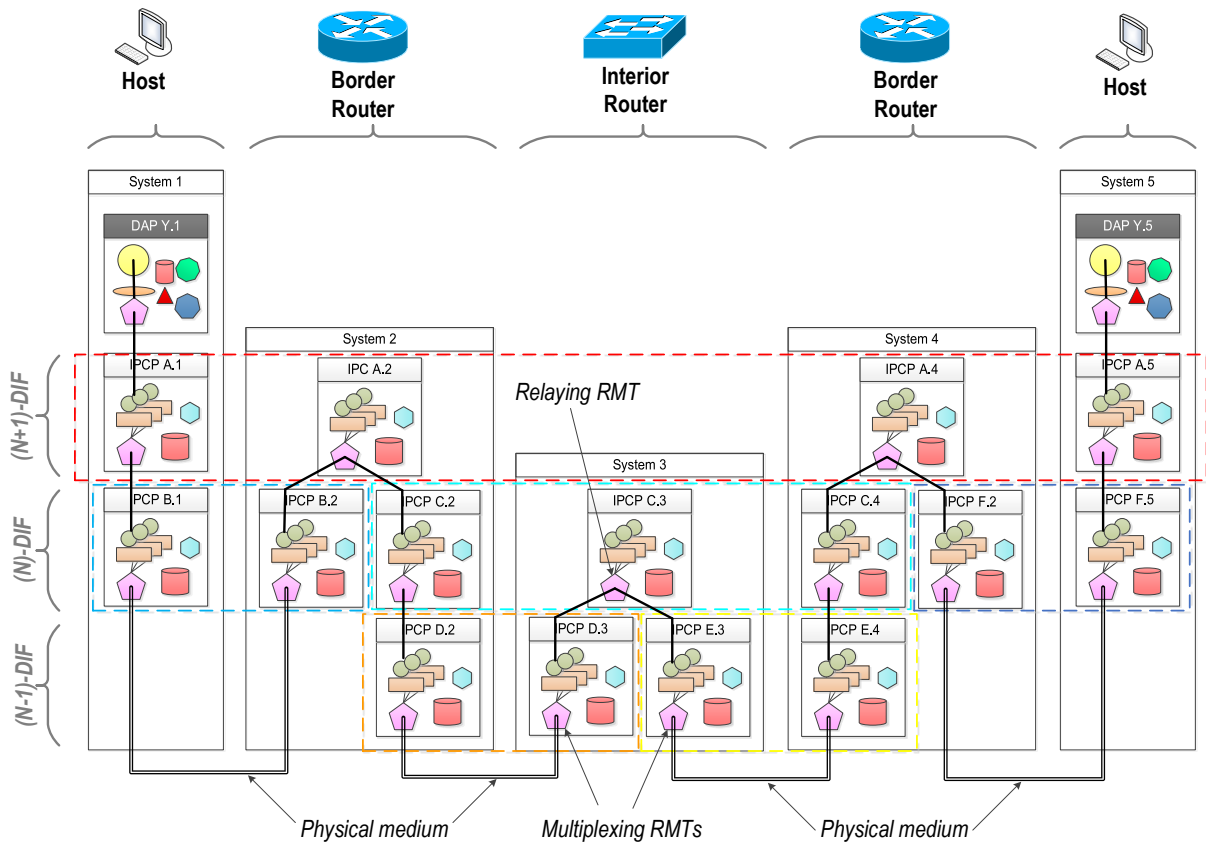


Fig. 52: Example of RINA network with three levels of DIFs and different nodes

As seen in Fig. 52, the main difference between node kinds is in an overall number of DIF levels present in a computing system. Due to the limited number of network interface cards (NIC), Hosts usually have a single 0-DIF (connected to the physical medium) and a few 1-DIFs leveraging on this lowest level DIF. Interior routers have potentially a lot of 0-DIFs (for each interface) but only a few relaying 1-DIFs. Border routers also perform relaying but serve as gateways between those (N-1)-DIFs, which are not connected directly. Thus, (N-2)-DIF is needed to reach physical medium.

5.2.2 Distributed Application Process Components

IPC Management is an integral part of any DAP responsible for managing supporting DIFs and providing their services to participating APs. IPC Management consists of following components depicted in Fig. 53:

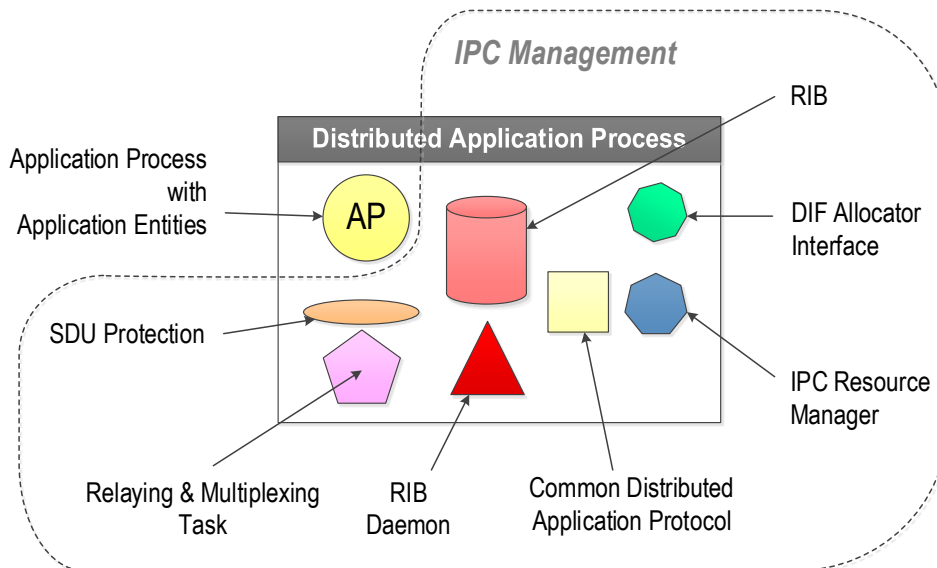


Fig. 53: Distributed Application Process components

Only IPC Resource Manager and DIF Allocator interface are exclusive to IPC Management, other components are also present in IPC Process and described later.

DIF Allocator

The primary task of **DIF Allocator (DA)** is to return a list of DIFs where destination application may be found given ANI and access control information. Additional and more complex DA description is available in [143]. DA contains and works with multiple mapping tables to provide its services:

- **Naming information table** – provides association between APN and its synonyms;
- **Search table** – provides mapping between requested APN and the list of DAs where to search for it next;
- **Neighbor table** – maintains a list of adjacent peers when trying to reach other DAs;

- **Directory** – contains records mapping APNs with access rights to the list of supporting DIFs including DIF’s name, access control information and provided QoS.

IPC Resource Manager

IPC Resource Manager (IRM) (see specification [144]) as its name suggests manages DAF resources.

This involves multiple different tasks:

- IRM processes *allocate* calls by delegating them to appropriate local IPCPs in relevant DIFs;
- IRM manages DA queries and acts upon their responses. When the DA response contains more than one DIF, IRM chooses which DIF to use;
- IRM administers the use of flows between AEs and DIFs. IRM may choose to multiplex a single or multiple AE flows into a single/multiple flows to a set of DIFs;
- IRM initiates joining or creating DAF and/or DIF. IRM acts upon the DAF, or DIF lost (e.g., sending notifications or perform subsequent actions).

5.2.3 IPC Process Components

IPC Process is instance within DIF, which allows the computing system to do IPC with other DIF members. Each IPC process performs (secure/reliable) data transport, (authenticated) enrollment, (de)allocation of resources, routing, management and more. Functions could be categorized under one of following categories: a) data transfer; b) data transfer control; and c) IPC management. Each category with different processing timescale and complexity – a) is simplest and performed the most often, c) the least often but the functionality is rather complex.

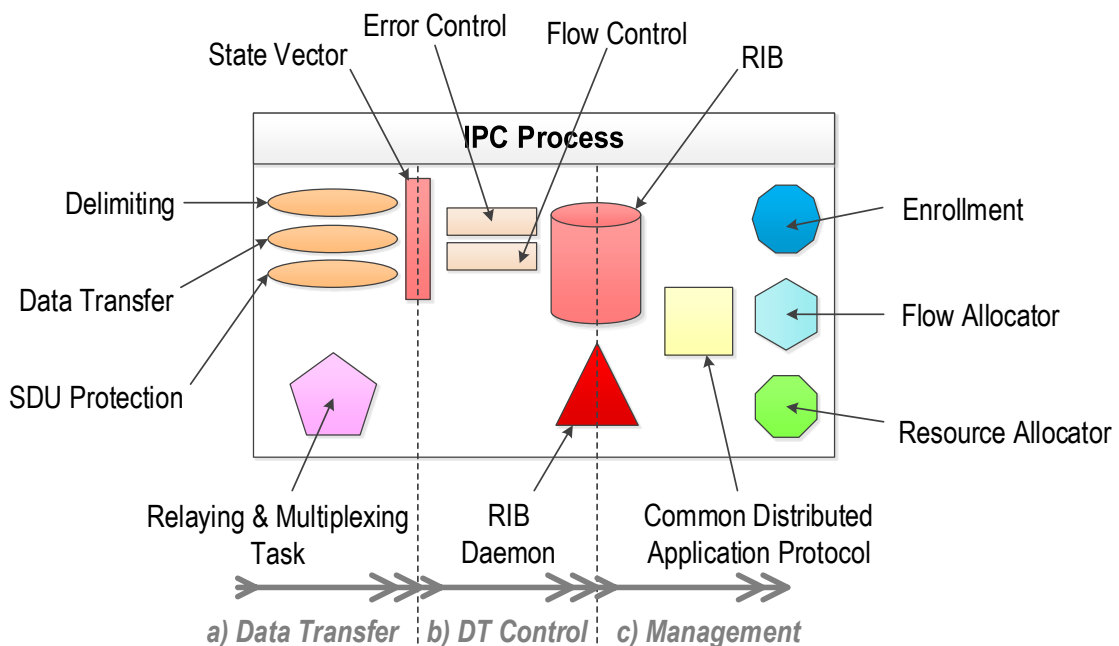


Fig. 54: IPC Process components

IPC provides API to a DIF/DAF above, which requested its service. Basic **IPC API** offers four operations: *allocate* (allocates communication resources); *deallocate* (releases previously allocated resources); *send* (passes SDU to IPC) and *receive* (retrieves SDU from IPC). Calls may be further subdifferentiated as *allocate request*, *allocate response*, *deallocate submit* and *deallocate deliver*.

Graphical representation of IPC Process and its most important components is depicted in Fig. 54. A brief description of each component and their functionality is provided below figure. Some components outlined below also contain policy descriptions. Those policies are mentioned because they are relevant to our contribution. The complete list of current policies with a brief info is in Addendum 8.5.

Enrollment

Enrollment takes place whenever IPCP joins existing DIF. IPCP newcomer creates a connection with another IPCP (which is already a member) allocating (N-1)-flow. It is then authenticated to whatever degree required by the policy. Enrollment occurs after successful connection establishment. Enrollment procedure of a new member should be dependent on a connection use-case. For instance, there may be a different exchange of messages for: a) the new member joining DIF for the first time; b) the IPCP that had been already a member of DIF and right now is rejoining. The new member either tells or gets its address to/from a DIF. Enrollment procedure is codified in [145].

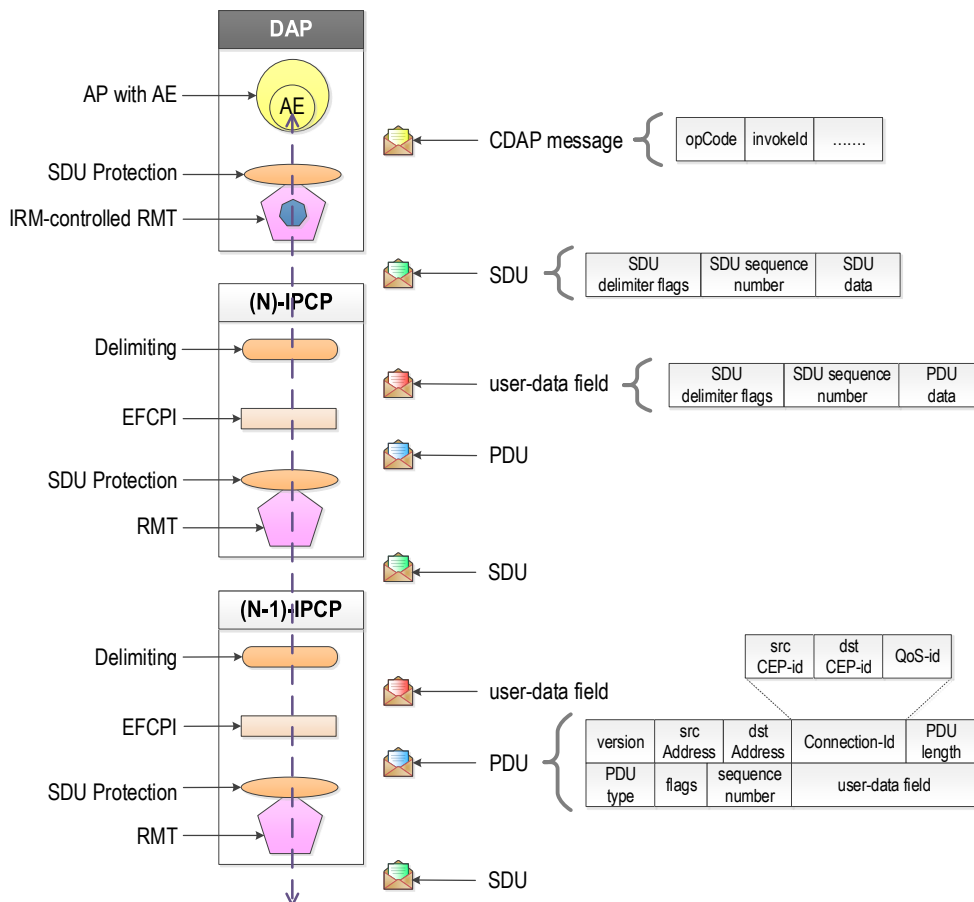


Fig. 55: Message passing between RINA components

Delimiting

SDU in RINA is a unit of data that is delivered as a whole at the destination. IPC might fragment SDU (when passing it down) or re-assemble user-data (when passing it up). Hence, the operation performed by **Delimiting** module (for specification see [146] and [147]) is to delimit SDU into/from PDU's user-data preserving its identity. Employed mechanism indicates the beginning and/or the end of SDUs. Either internal (special pattern) or external (SDU length in PCI) delimiting could be used.

Encapsulation/Decapsulation of data messages happens in RINA components lying in the data path. Fig. 55 depicts this process DIF/DAF together with messages nomenclature.

Data Transfer with Error/Flow Control

Error and Flow Control Protocol (EFCP) is split into two independent PMs coupled and coordinated through a state vector. As EFCP name suggests, EFCP guarantees data transfer and data control. Full EFCP functionality is described in [148]. However, these specifications are currently being revisited.

Data Transfer Protocol (DTP) implements mechanisms tightly coupled with transported SDUs, e.g., fragmentation, reassembly, sequencing. DTP PM operates on a data PDU's PCI with fields requiring minimal processing – source/destination addresses, QoS requirements, Connection-id, optionally sequence number or checksum. DTP carries user-data.

Data Transfer Control Protocol (DTCP) implements mechanisms that are loosely coupled with transported SDUs, e.g., (re)transmission control using various acknowledgment schemes and flow control with data-rate limiting. DTCP functionality is based on Watson's Delta-t and DTCP PM processes control PDUs. DTCP provides error and flow control over user-data.

There is **EFCP instance (EFCPI)** module per every active flow. EFCPI consists of DTP and DTCP submodules. DTCP policies are driven by the quality of service demands. DTCP submodule is unnecessary for flows that do not need it, i.e., flows without any requirements for reliability. The relationship between DTP and DTCP is illustrated in Fig. 56. Depicted are also data transfer and data control transfer paths. Control traffic stays out of the main data transfer.

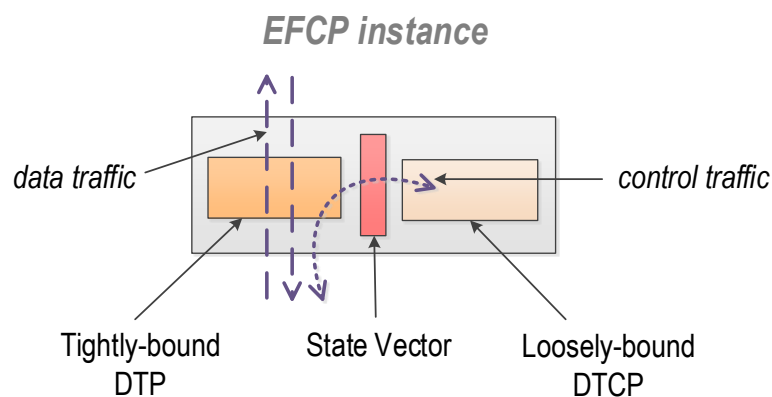


Fig. 56: EFCP instance divided into DTP and DTCP part

Relaying and Multiplexing Task

Relaying and Multiplexing Task (RMT) modules have two primary responsibilities – relaying and multiplexing as characterized in [149]. The goal of multiplexing is to pass PDUs from EFCPIs and RIB Daemon to appropriate (N-1)-flows and reverse of that. Relaying handles incoming PDUs from (N-1)-ports that are not directed to its IPCP and forwards them to other (N-1)-ports using the information provided by its forwarding policy.

RMT instances in hosts and bottom layers of routers usually perform just the multiplexing task, while RMTs in top layers of interior/border routers do both multiplexing and relaying. In addition to that, RMTs in top layers of border routers perform flow aggregation. Primary RMT functions are demonstrated in Fig. 52.

Each (N-1)-port handled by RMT has its set of input and output buffers. The number of buffers, their monitoring, their scheduling discipline and classification of traffic into separate buffers are all matter of policies.

RMT is a straightforward high-speed component. As such, most of its management (state configuration, forwarding policy input, buffer allocation, and data rate regulation) is handled by the Resource Allocator, which makes the decisions based on observed IPC process performance.

Each IPC process has to solve the forwarding problem: given a set of EFCP PDUs and (N-1)-flows leading to various destinations, to which flow should be each PDU forwarded? In RINA, the decision is handled by the RMT and its *PDUForwardingPolicy*. The *PDUForwardingPolicy* may consist of looking up the PDU's destination in its forwarding table (resembling the forwarding mechanism in traditional TCP/IP routers). When in need of deciding for an output (N-1)-port for a PDU, the *PDUForwardingPolicy* is given the PDU's PCI and then it returns a set of (N-1)-ports to which the PDU has to be sent. This provides enough granularity to implement multiple communication schemes apart from unicast (such as multicast or load-balancing) because the decision is left to the *PDUForwardingPolicy*. E.g., a simple forwarding policy would return a single (N-1)-port based on PDU's destination address and QoS-id, whereas in case of a load-spreading policy and multiple (N-1)-ports leading to the same destination, the policy could split traffic by PDUs' flow-ids and always return a single (N-1)-port from the set.

SDU Protection

SDU Protection is the last part of the IPCP data path, before an SDU is handed over to an underlying DIF. It is responsible for protecting SDUs from untrusted (N-1)-DIFs by providing mechanisms for lifetime limiting, error checking, data integrity protection and data encryption. It also provides mechanisms for data compression and a potential placeholder for other two-way manipulations.

SDU Protection handles each (N-1)-flow separately due to different levels of trust. This gives SDU Protection the ability to skip some mechanisms in favor of performance for trusted networks while still being protected from untrusted networks. Therefore, SDU Protection employs various policies, e.g:

a) *NullSDUProtection* that performs no transformations; b) *BasicSDUProtection* that applies life time limiting and error checking; c) *CryptographicSDUProtection* that extends the *BasicSDUProtection* by adding cryptographic encryption of data and an integrity check using a cryptographic hash of the content.

Flow Allocator

Flow Allocator (FA) processes *allocate/deallocate* IPCP API calls and further management of all IPCP's flows. FA instantiates a Flow Allocator Instance to manage each flow; FA is controller/container for all Flow Allocator Instances.

Flow Allocator Instance (FAI) is created upon *allocate request* call, and it manages a given flow for its whole lifetime. FAI handles creating/deleting EFCPI(s) while managing a single flow's connection. FAI returns port-id to the allocation requestor upon successful allocation as a referencing handle. FAI participates only on port allocation, not on synchronization, which is the responsibility of EFCPI. The FAI maintains a mapping between flow's local port-id and connection's local CEP-id.

FA contains **Namespace Management (NSM)** interface for assigning and resolving names (including synonyms) within DIF. This activity involves maintaining the table with entries that map requested ANI to IPCP's address.

Flow object contains all information necessary to manage any given flow between communicating parties. It is carried inside *create/delete flow request/response* messages controlling FA and FAI operation. Flow object contains: source and destination ANI, source and destination port-ids, connection-id, source and destination address, QoS requirements, a set of policies, access control information, hop-count, current and maximal retries of *create flow requests*.

Flow allocation processes for (N)-DIF between two APs on different systems is depicted in Fig. 57. It assumes that relevant (N-1)-flows have been already allocated using the same principle as the one being described but on different DIF's rank.

- #1) *API* issues *allocate request* that is delivered to IPCP A.1. If it is valid and well-formed then it spawns FAI to manage requested flow. FAI resolves AP3's APN to one of DIF A addresses (A.3). It instantiates EFCPI (with CEP-id) and creates bindings between EFCPI and RMT. *Create flow request* is sent as the last step;
- #2) *Create flow request* arrives at "System 2". IPCP A.2's FA processes the request and discuss NMS. It discovers that request is not intended for any local AP. FA looks up the destination discovering that A.3 should be a next-hop. FA forwards the request to "System 3";
- #3) The request arrives at IPCP A.3. Over there, FA determines by querying NMS that *create flow request* destination address is its address. Thus, destination AP resides on this system. FAI is spawned and determine whether the request can be accommodated. If not then negative *create flow response* is sent back to the requestor. Otherwise, FAI notifies destination AP with *allocate request*;

- #4) If destination AP accepts or rejects the request then either positive, or negative *allocate response* is returned to FAI. Based on the response, FAI binds port-id, instantiates EFCPI, creates bindings. Flow object is updated (with local port-id and CEP-id) and sent back as positive/negative *create flow response*. Response is just relayed (not processed) on interior routers (IPCP A.2);
- #5) Originating *A.I*'s FAI receives *create flow response* and updates relevant flow object. If the response is positive, then, FAI notifies source AP with positive *allocate response* and APs may commence data transfer. If the response is negative, then FAI invokes retry policy to correct flow creation or deal appropriately with failure (i.e., passing negative *allocate response*).

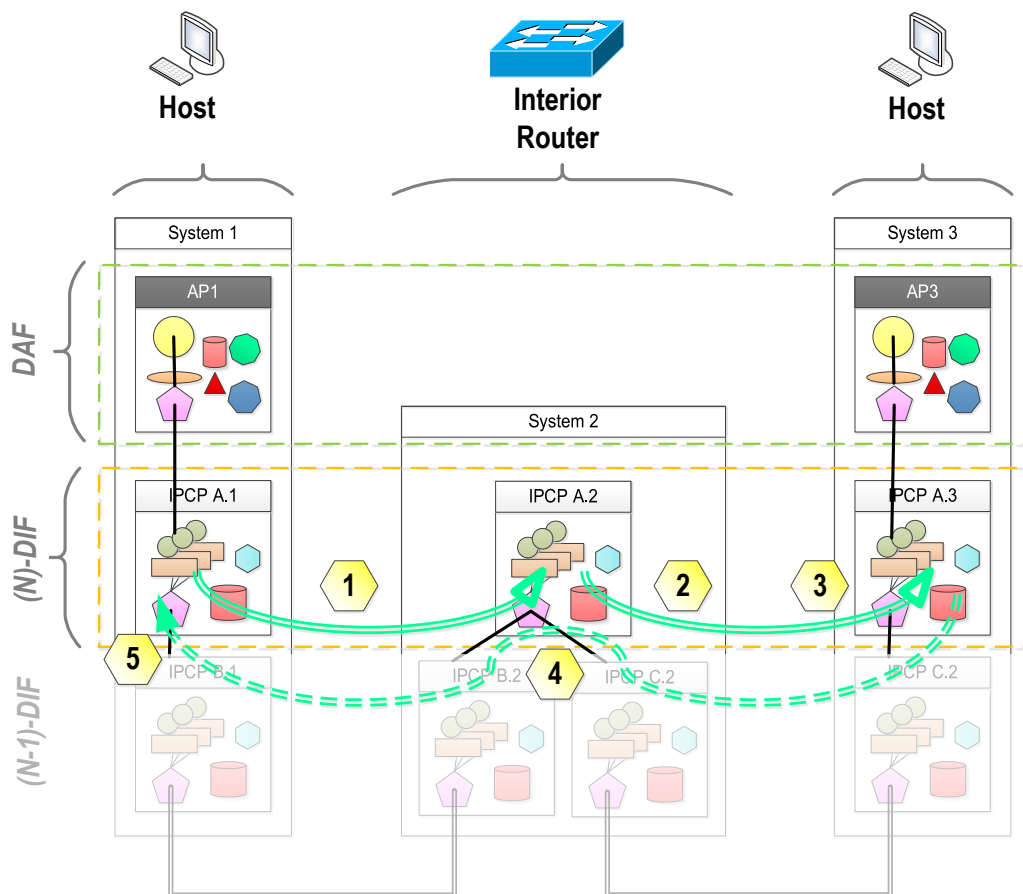


Fig. 57: Flow allocation process

Original specification [150] were refined as the subject of this thesis contribution. Detail description of flow allocation and deallocation is provided in Figures Fig. 58, Fig. 59, Fig. 60 and Fig. 61. Transitions are denoted with “input / action” labels. FA and FAI maintain state for any given flow and refuse inappropriate transitions (e.g., initiating deallocation before the allocation is successful). These transitions are omitted for clarity. There are four different FSMs. Fig. 58 depicts FA operation reacting upon notification from RIBd. Fig. 59 and Fig. 60 show flow allocation procedure for initiating and responding FAIs. Fig. 61 illustrates flow’s lifecycle after successful allocation, and it is mutual for both initiating and responding FAIs.

NewFlowRequestPolicy is invoked after FAI's instantiation. Policy subtasks involve both 1) evaluation of access control rights; and 2) translation of QoS requirements specified in *allocate request* to appropriate RA's QoS-cubes. *AllocateRetryPolicy* occurs whenever initiating FAI receives negative *create flow response*. This policy allows FAI to reformulate the request and/or to recover properly from failure. *AllocateNotifyPolicy* controls a proper time when source AP is going to be notified of the result of allocation by initiating FAI. It may be either when EFCPI is created, or when allocation is confirmed by destination or any other notification strategy may be employed. *SeqRollOverPolicy* is invoked simultaneously by both initiating and responding FAIs whenever PDU's sequence number threshold is reached. The policy usually spawns new EFCPIs and changes bindings.

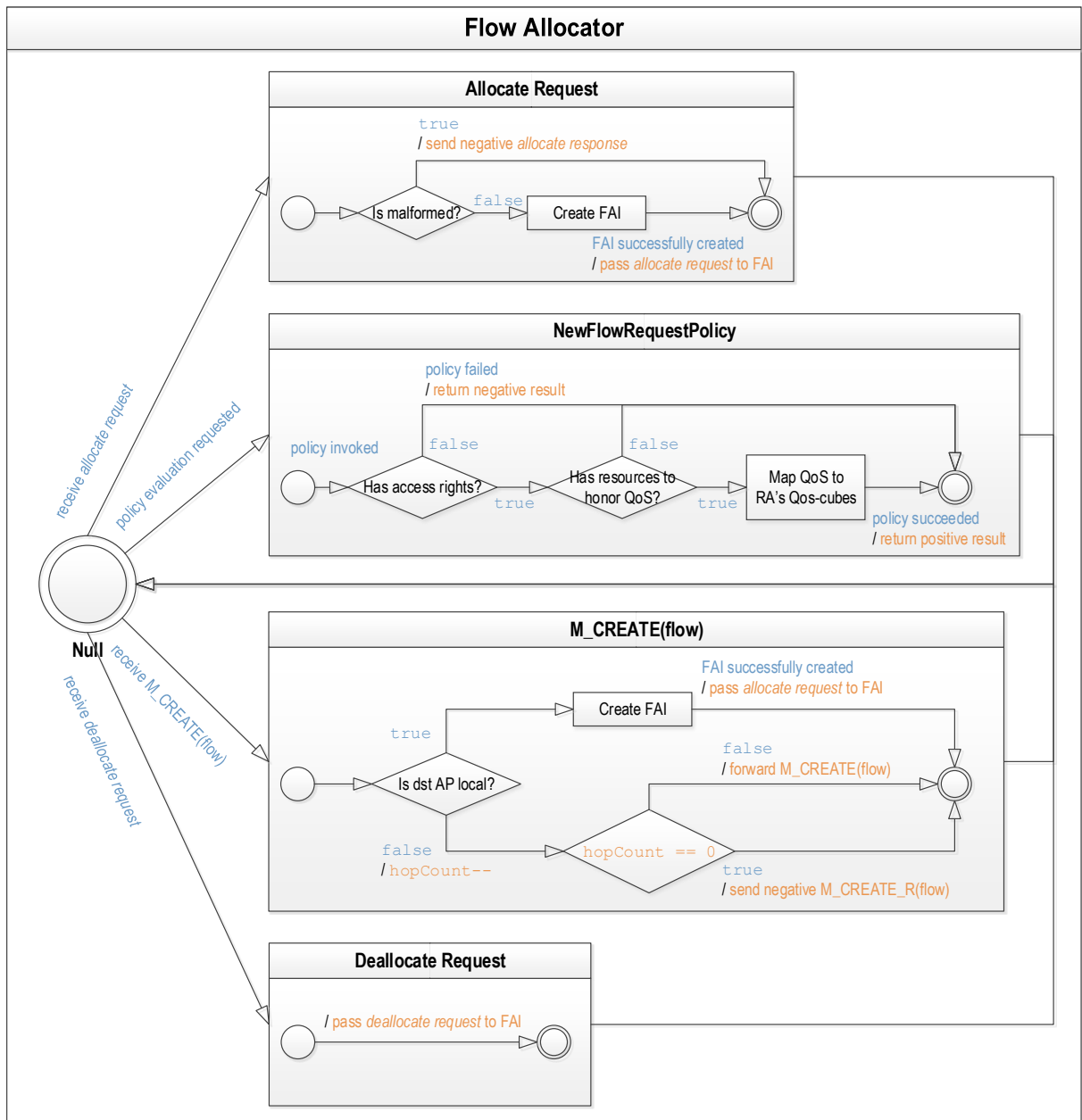


Fig. 58: Flow Allocator operation

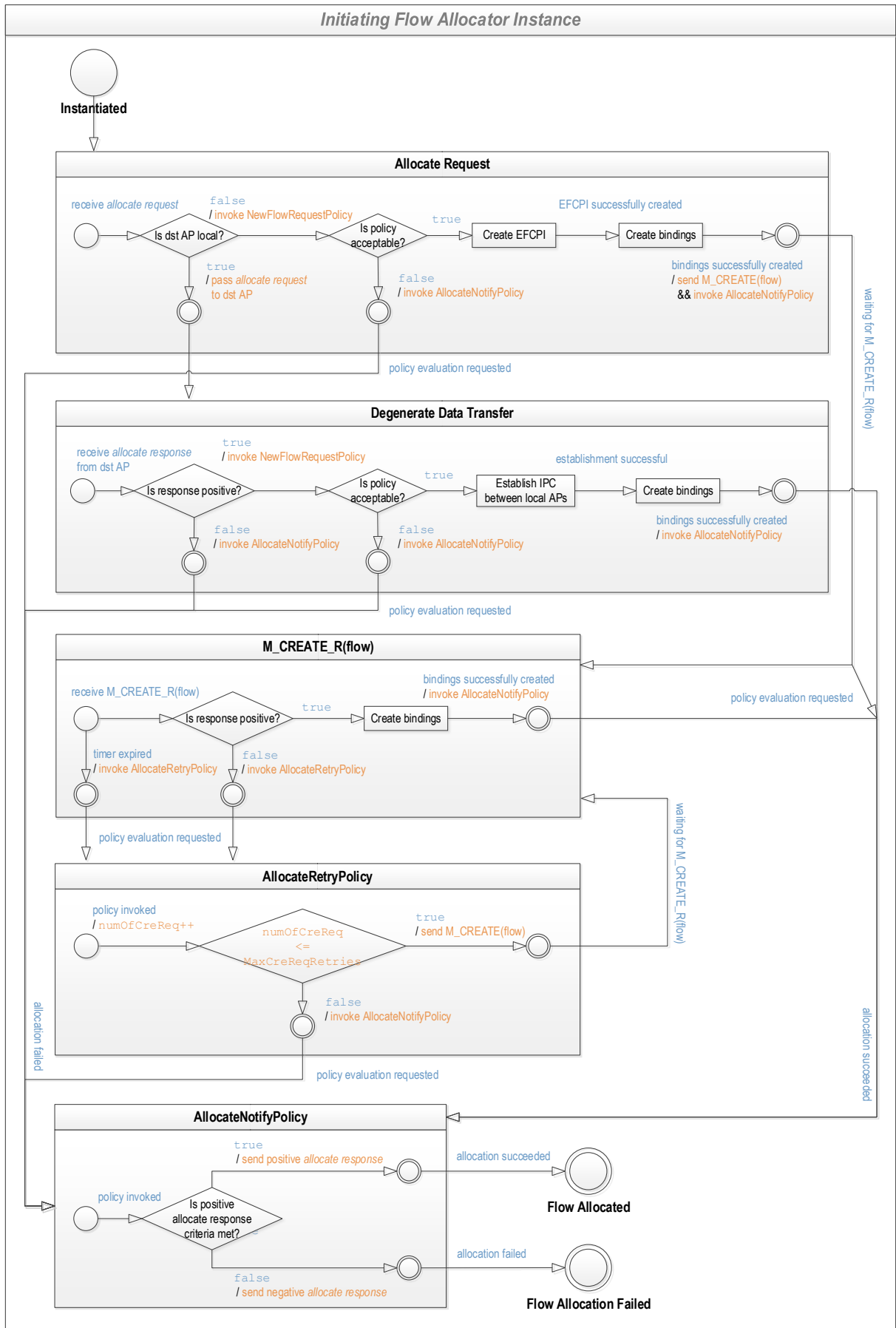


Fig. 59: Flow Allocator Instance operation of initiating IPCP

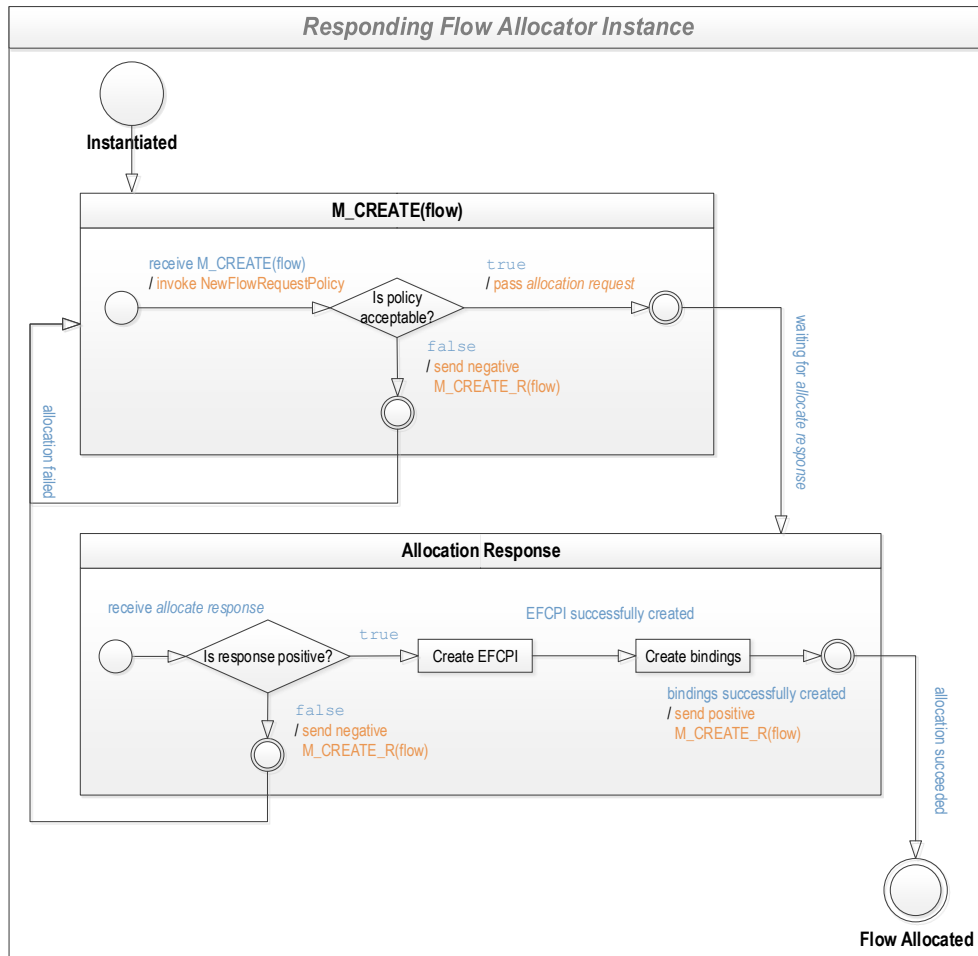


Fig. 60: Flow Allocator Instance operation of responding IPCP before the flow was allocated

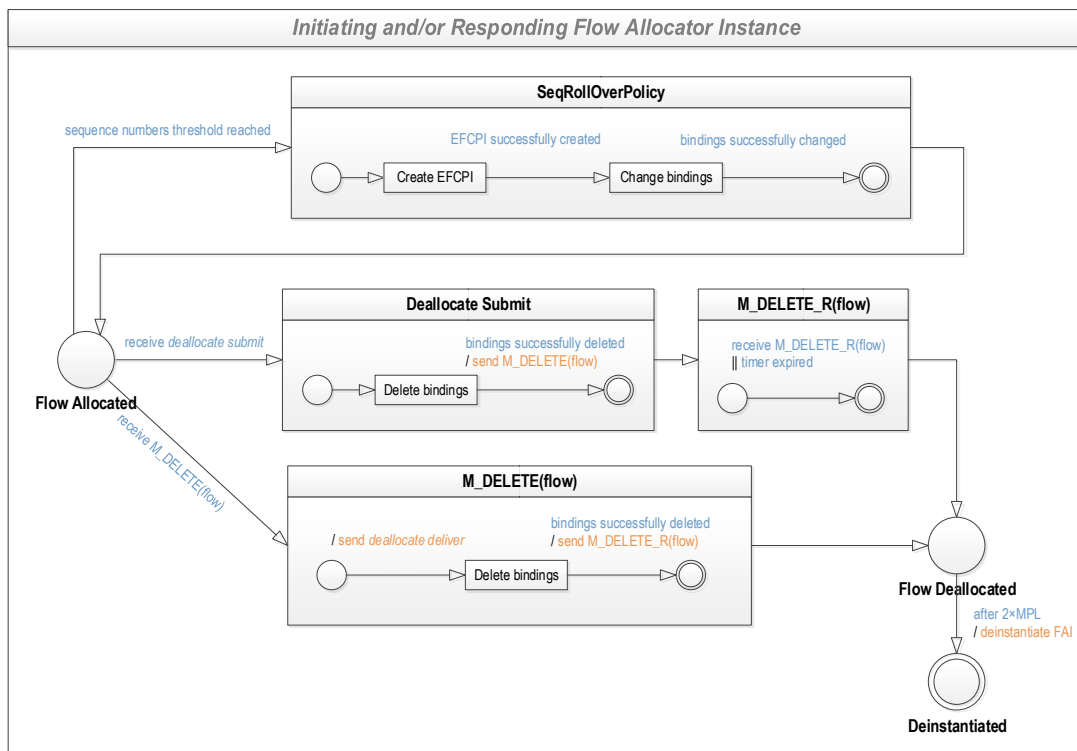


Fig. 61: Flow Allocator Instance operation after the flow was allocated

Resource Allocator

If a DIF has to support different qualities of service, then different flows will have to be allocated to different policies and traffic for them treated differently. **Resource Allocator (RA)** delineated in [151] is a component accomplishing this goal by handling management of various IPCP resources, namely it:

- controls creating/deleting and enlarging/shrinking of RMT queues;
- modifies EFCPI's DTCP policy parameters;
- controls creating/deleting of (N-1)-flows and their assignment to appropriate RMT queue(s);
- manages QoS classes and their assignment to RMT queue(s);
- manages routing information affecting RMT's relaying or initiates congestion control.

RA maintains a catalog of meters and dials by monitoring various management resources. Each catalog item can be manipulated and shared with other IPC processes within DIF.

Generating information necessary for *PDUForwardingPolicy* is one of the tasks of RA, namely its subcomponent called **PDU Forwarding Table Generator**. For this purpose, RA uses pieces of information provided by other sources, most notably the *RoutingPolicy*.

The *RoutingPolicy* exchanges information with other IPCPs in the DIF in order to generate a next-hop table for each PDU (usually based on the destination address and the id of the QoS class the PDU belongs to). The next-hop table is then converted into a **PDU Forwarding Table** with input from the PDU Forwarding Table Generator, by selecting an N-1 flow for each "next-hop". *RoutingPolicy* may resemble distance vector and link-state routing protocols used in today's Internet, but the current research is also aimed at other paradigms such as topological/hierarchical routing, greedy routing or MANET-like routing.

RIB Daemon

All information maintained by IPC tasks such as FA, RA, and others is available and updated through **RIB Daemon (RIBd)** described in [152] and [153]. Information exchange is necessary to coordinate the distributed IPC. Different update strategies for different kinds of information may be used to synchronize state between different DIF member subsets.

Resource Information Base (RIB) is a logical database of information accessible via RIB Daemon. By logical database, we mean that some of RIB information may be stored in the dedicated database and the rest of IPCP components. Periodic or solicited events can cause RIB to be queried/updated by IPCP peers via management CDAP messages (e.g., routing updates). RIBd provides an API to perform an operation on both local and remote RIB.

Common Distributed Application Protocol

Subsection 5.1.1 postulates that there is only a single application protocol required and this is the **Common Distributed Application Protocol (CDAP)**. DIFs use CDAP for all non-data communication

(i.e., IPC management such as maintaining RIB, controlling flow allocation, joining a DIF). DAFs may not use CDAP for backward compatibility. However, CDAP expressiveness should allow the transition of legacy protocols. CDAP is based and patterned on two existing protocols – ACSE (see [154] and [155]) for the establishment phase, CMIP [156] for the data transfer phase.

Establishment subpart is called out separately (for legacy protocols it may be used as a wrapper providing authentication service). Data transfer subpart is object-oriented (with built-in scope and filter support) protocol offering six primitive operations: *create*; *delete*; *read* (i.e., get value); *write* (i.e., put or set value); *start* (i.e., execute action) and *stop* (i.e., suspend action). The collection of objects is dependent on used AE, which provides access rights to them.

CDAP has modular structure composed of three submodules to provide flexibility:

- The common application connection establishment (CACE) submodule;
- The authentication (Auth) submodule provides authentication of the communication endpoints. A range of submodules will be available to support different kinds (e.g., none authentication, shared password, certificates) of authentication policies employing different cryptographic tools (e.g., a-/symmetric ciphers for confidentiality, MAC codes for integrity);
- The CDAP submodule.

CDAP offers following eighteen message types summarized in Tab. 13 [157]:

Opcode	Description
<i>M_CONNECT</i>	Initiate a connection from a source application to a destination application
<i>M_CONNECT_R</i>	Response to <i>M_CONNECT</i> carries connection information or an error indication
<i>M_RELEASE</i>	Orderly close of a connection
<i>M_RELEASE_R</i>	Response to <i>M_RELEASE</i> carries final resolution of close operation
<i>M_CREATE</i>	Create an application object
<i>M_CREATE_R</i>	Response to <i>M_CREATE</i> carries result of creating request, including identification of the created object
<i>M_DELETE</i>	Delete a specified application object
<i>M_DELETE_R</i>	Response to <i>M_DELETE</i> carries result of deletion attempt
<i>M_READ</i>	Read the value of a specified application object
<i>M_READ_R</i>	Response to <i>M_READ</i> carries part or all of object value or error indication
<i>M_CANCELREAD</i>	Cancel a prior read issued using <i>M_READ</i> for which a value has not been completely returned
<i>M_CANCELREAD_R</i>	Response to <i>M_CANCELREAD</i> indicates outcome of cancelation
<i>M_WRITE</i>	Write a specified value to a specified application object
<i>M_WRITE_R</i>	Response to <i>M_WRITE</i> carries result of write operation
<i>M_START</i>	Start the operation of a specified application object, used when the object has operational and non-operational states
<i>M_START_R</i>	Response to <i>M_START</i> indicates the result of the operation
<i>M_STOP</i>	Stop the operation of a specified application object, used when the object has operational and non-operational states
<i>M_STOP_R</i>	Response to <i>M_STOP</i> indicates the result of the operation

Tab. 13: CDAP message types

Connection management between two applications is divided into two traditional phases – establishment and data transfer. An AP issues *allocate request* to underlying DIF’s IPCP specifying the destination APN and QoS requirements. If the allocation is successful, IPCP returns port-id to be used as a handle for all communication leveraging this flow. When the previous phase is completed, CACE sends a *M_CONNECT* message to start authentication using Auth submodule. Additional message exchange might follow in order to support different authentication mechanisms. If it is successful then the connection is established and CDAP transits to data transfer phase.

Another contribution is further refinement of CACE specifications [158]. Detail description of CDAP operation is provided in Figures Fig. 62, Fig. 63 and Fig. 64. Once again transitions are denoted with “input / action” labels. There are three different FSMs. Fig. 62 depicts establishment phase on initiating the process. Fig. 63 shows the same but from the perspective of the responding process. Fig. 64 outlines data transfer phase for both initiator and responder once they successfully reach “Established“. For the sake of readability, only correct transitions are shown. Incorrect transitions upon receiving unexpected CDAP message terminate from any state in “Error” marked as “wrong input”. Both initiator and responder might “indicate deallocation”, thus entering “Deallocating” state at any given moment.

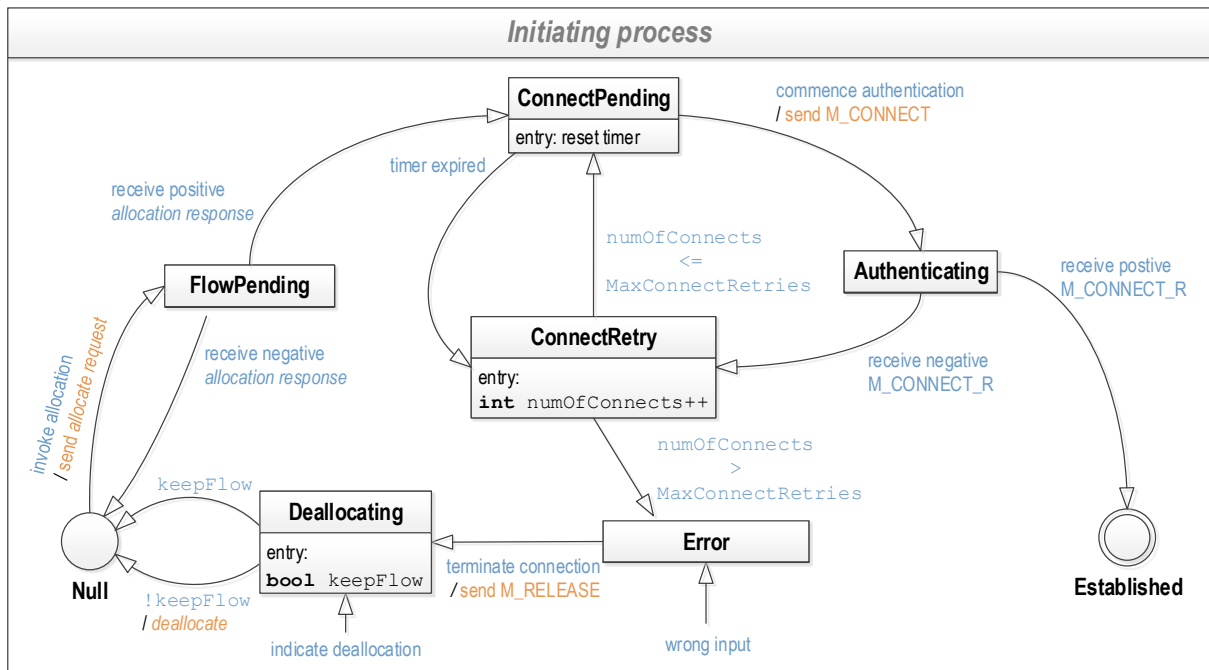


Fig. 62: Establishment phase on initiating process

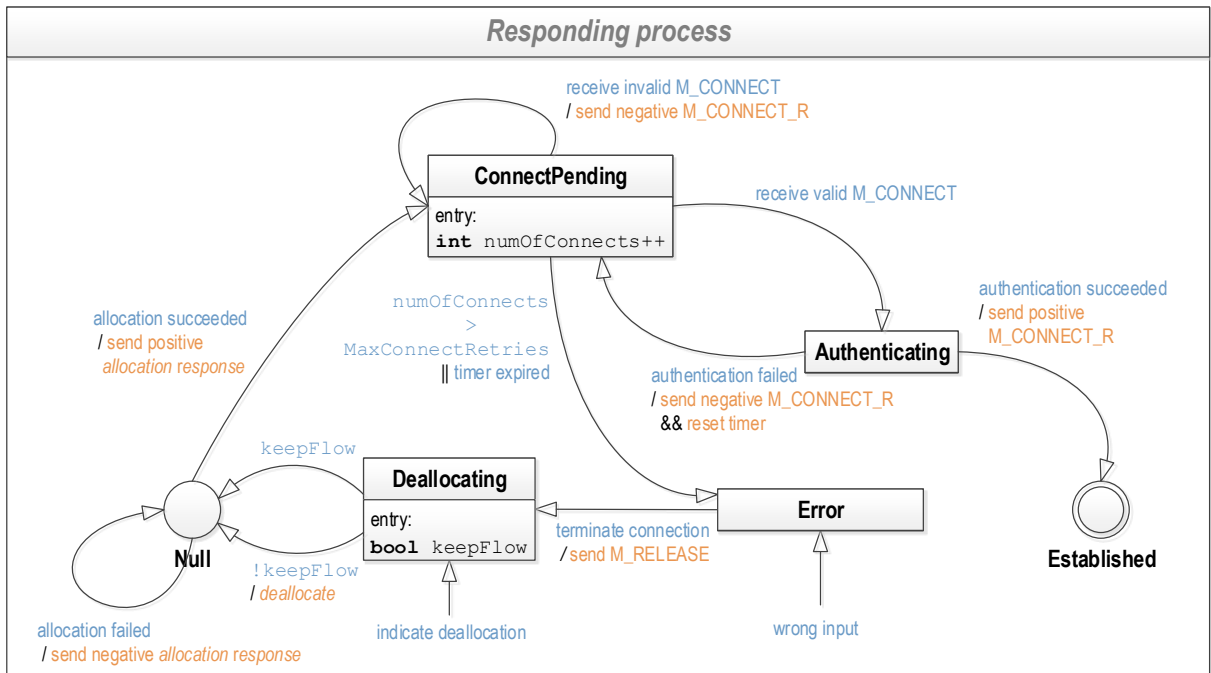


Fig. 63: Establishment phase on responding process

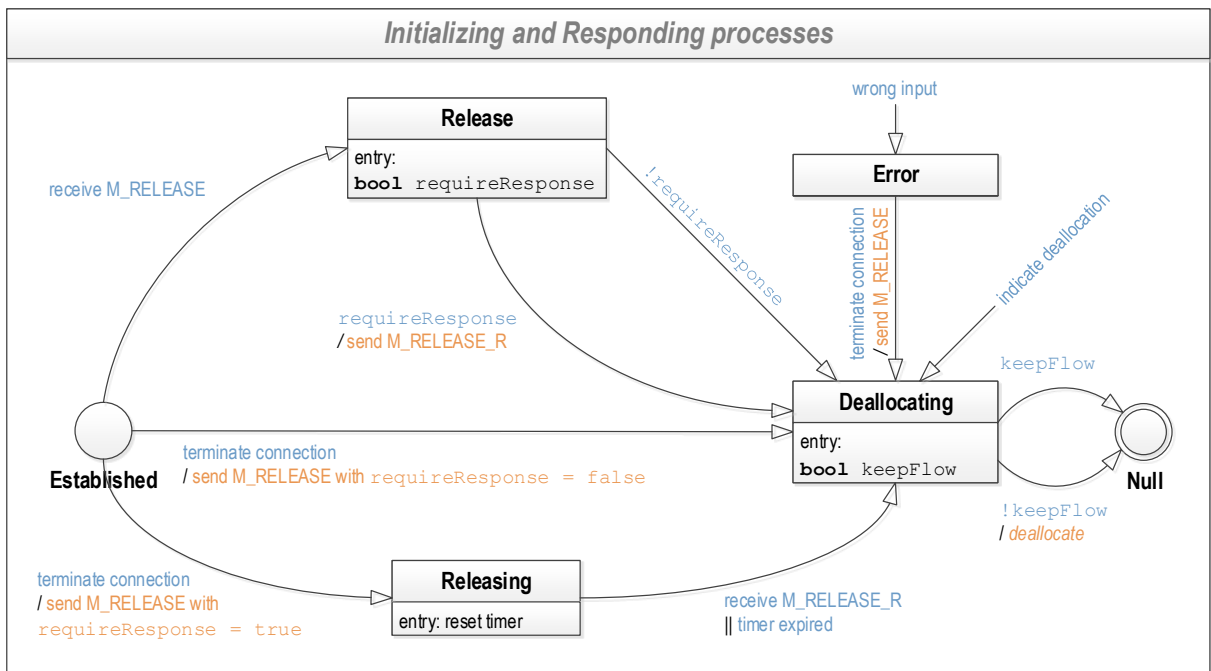


Fig. 64: Data transfer phase on initiating/responding process

Depending on whether (N-1)-flow should be preserved or not, the transition from “Deallocating” (based on keepFlow boolean) may delete any state associated with connection and transit to the “Null” state.

5.3 State-of-the-Art

This subchapter mentions coordinated research activities concerning RINA. Moreover, notable implementations are introduced and facts about RINA readiness and deployment status.

5.3.1 Projects

RINA is successfully targeted in the frame of multiple EU projects as an alternative to traditional TCP/IP stack. Here is a list of projects and their main interests concerning RINA:

- IRATI [159] – IRATI advances the state-of-the-art of RINA towards an architecture reference model and specifications that are closer to enable implementations deployable in production scenarios. The design and implementation of IRATI prototype on top of Ethernet permits further evaluation and deployment of RINA in real computer networks;
- IRINA [160] – IRINA aims to compare RINA against TCP/IP in a lab environment using IRATI prototype. Moreover, it proposes use-cases, where RINA is better option for NREN⁷⁰ scenarios;
- PRISTINE [161] – PRISTINE investigates programmability of RINA architecture, namely its separation of mechanisms and policies to achieve more flexible behavior of network components;

5.3.2 Implementations

IRATI Stack

IRATI [162] is an open source network stack implementation of the RINA targeted to the OS/Linux system written in C. It consists of a kernel (packet handling) and user-space (IPCP configuration) parts. Currently RINA stack may operate either over TCP (using port numbers) or directly over Ethernet (using VLAN tags) employing Shim DIFs (see [163] and [164]). The main component is IPC Manager that handles creation/destroying of IPCPs and governs flow allocation.

ProtoRINA

ProtoRINA [165] is a Boston University's RINA user-space prototype written in Java. ProtoRINA provides a limited framework for experimenting with RINA concepts within GENI⁷¹ testbed (see [166]).

5.3.3 Simulators

We are not aware of any existing discrete-event simulator that could be used for research or educational tool. None, apart from our own RINASim exhaustively described in the following section.

⁷⁰ **National research and education network (NREN)**: E.g., Czech CESNET or European GEANT. For more, see https://en.wikipedia.org/wiki/National_research_and_education_network

⁷¹ **Global Environment for Network Innovations (GENI)**. For more, see <http://www.geni.net>

5.4 Contribution

Simulation often serves for validating and verifying new technologies, which do not have a yet implementation. The simulation also finds weak points and drawbacks during test runs and subsequently allows one to enhance development process based on feedbacks. Hence, the implementation of the **RINA Simulator (RINASim)** is a natural step to support ongoing research and development of the Recursive Internet Architecture.

We are developing the RINASim in the frame of European project PRISTINE. RINASim is a stand-alone framework for OMNeT++ discrete event simulator environment. RINASim is coded from scratch and independent on another library. The main purpose is to offer the community with reliable and the most up-to-date tool (in the sense of RINA specification compliance) for simulating RINA-based computer networks. Thanks to the OMNeT++'s built-in result analysis and graphical simulation output, RINASim may be used not only for research but also as an educational tool.

This subchapter introduces RINASim installation guideline, development design and description of components interactions. Moreover, it illustrates RINA principles and RINASim functionality on one of the basic examples. Subchapter contains only the most relevant information due to the limited space, for more, please see PRISTINE deliverable 2.4 [167].

5.4.1 Installation

RINASim is developed in OMNeT++ 4.6, but its source codes are fully backward compatible with older OMNeT++ versions that support C++11 language standard and GCC 4.9.2 compiler. All source codes (including master and other thematic branches) are publicly available on the project's GitHub repository [168]. Apart from this official channel, RINASim stable release snapshots are periodically published on Open Source Project repository [169].

RINASim installation is a straightforward process with two phases: 1) importing the project into OMNeT++ IDE; 2) compiling the project, which creates one static library (`librinasimcore` containing simulation core) and one dynamic library (`librinasim` also containing various policies linked together with core).

5.4.2 Design

This subsection provides a general overview of RINASim components design, which includes high-level abstract models of computing systems (like hosts and routers) and also their low-level submodules (like IPCP). In general, a structure of RINASim models follows the structure proposed in the RINA specification. This intentional correspondence enables anyone understanding the RINA specifications to easily orient in RINASim too. Though this structure does not always stand for the most natural representation of RINA concepts in simulation models, it provides a framework for evaluating properties

of the architecture and to identify missing or inaccurate information in the original specification. During the design of simulation models, we were able to identify several places where specifications should be refined to provide complete and unambiguous information. Following lines reflect RINASim design relevant to a date of this thesis.

Computing System Modules

RINASim offers a variety of high-level models simulating the behavior of independent computing system. These models can be employed to set quickly up simulation experiments. Through parameterization and extension, it is possible to test different deployments and settings. Based on the RINA specifications, we can distinguish between the following node types:

- Host nodes, which represent devices or systems that run distributed applications. These nodes implement the full RINA stack and, also, contains an application process(es). AP instances are configured to communicate with each other to simulate the behavior of an arbitrary RINA application. Currently, there are several predefined host nodes depending on a number of APs and AEs. Fig. 65 illustrates some of host nodes internal structure. The most of depicted hosts contain two IPCPs, which models usual end-system with a single NIC. The host may contain only single IPCPs, which would allow IPC with only one directly connected neighbor. Alternatively, host may contain more than two IPCPs; (0)-rank IPCPs represent multiple NICs, and (1+)-rank IPCPs represent different DIFs host memberships;

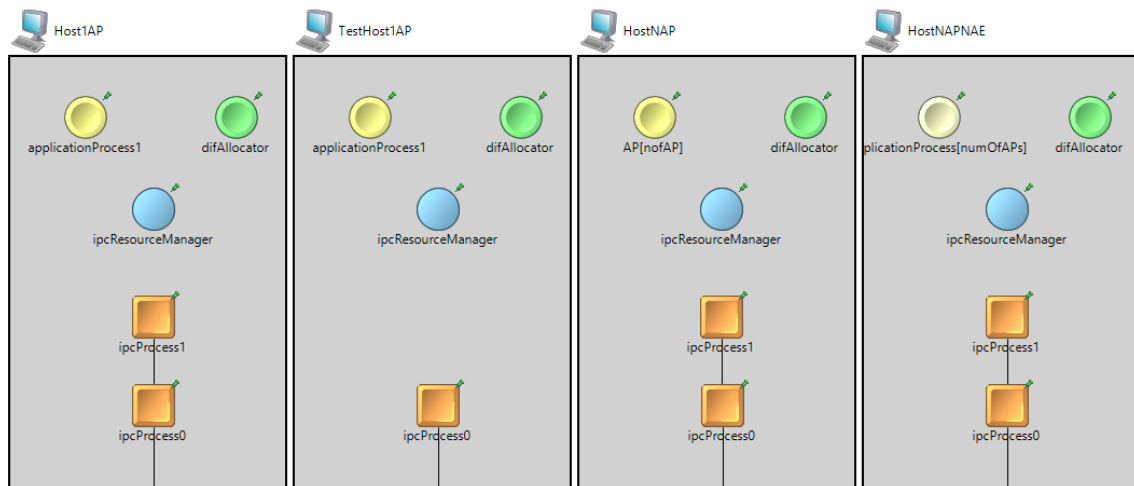


Fig. 65: Host nodes structure examples

- Routers (intermediate nodes), which can be either interior or border. A router is a device that interconnects different underlying DIFs and often does not run user applications. Just as in RINA specification, there are either interior or border routers depending on DIF stack depth (influenced partially also by a number of interfaces). Fig. 66 illustrates two interior routers and one border router simulation models.

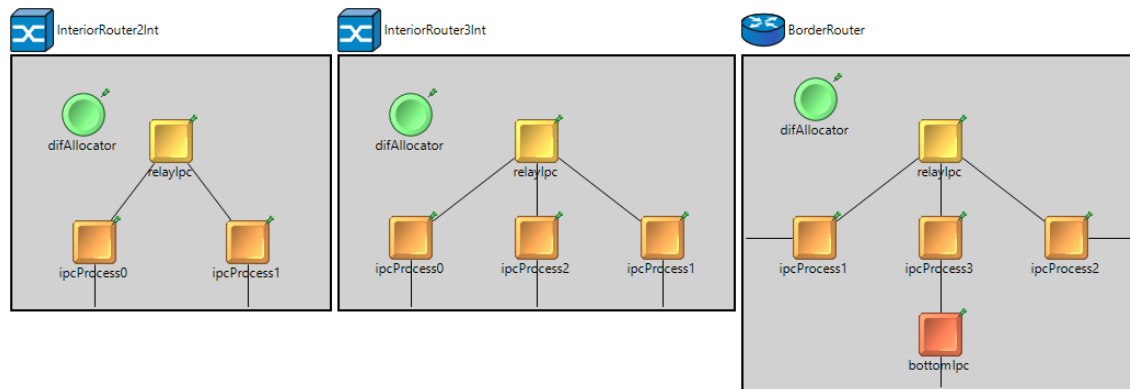


Fig. 66: Router nodes structure examples

Of course, there are many more possible combinations of host and router configurations than the ones currently defined in RINASim. However, the aim of providing predefined node models is not to cover all of the possible combinations but rather to offer the most used ones enabling to set quickly up simulation scenarios. Defining new node or router with suitable structure is not a complicated task. Nevertheless, the present collection of available models seems to be enough.

Policies

RINA specifications present the proposed network architecture as a generic framework, where mechanisms are intended to perform basic common functionality and policies are defined to select the most appropriate implementation of variable functionality. Rather than providing an exhaustive implementation of policies for each parameterized function, RINASim provides interfaces that are used by the core implementation to call functions defined by the selected policies.

The RINASim policy framework is based on OMNeT++ NED module interfaces [170], which helps to minimize the need for modifying existing C++/NED source codes. Instead of placing a simple module with a policy implementation inside the simulation network graph, a placeholder interface module is used. This design allows the potentially unlimited amount of user policy implementations to be defined and easily switchable via the configuration files (by setting a proper parameter of the encompassing module). Each policy consists of an NED module interface and a base C++ class. Fig. 68 shows an example of policy module interfaces (modules with “Policy” suffix in names) with loaded policies (blue labels above them).

DAF Modules

DAF components can be divided into three submodules: a) Application Processes (containing one or more Application Entities), which represents IPC endpoints; b) IPC Resource Manager, which interconnects APs and available IPCPs; c) DIF Allocator, which helps during APN discovery and management process. Components relationship and internal structure (described below) are depicted in Fig. 67.

The `applicationProcess` module contains `applicationEntity` submodules for each flow representing the connection between two applications. `applicationEntity` handles enforcing access control (by evaluating flow allocation requests), flow management and governing application protocol. Each `applicationEntity` contains `iae` (submodule interface, which allows pluggable change of application protocols) and the `commonDistributedApplicationProtocol` submodule that sends and receives messages on behalf of `applicationEntity`.

The `commonDistributedApplicationProtocol` submodule provides a simple object-based protocol for distributed applications. Currently, it is the part of RIBd and AE. CDAP is modeled as a compound module consisting of five main submodules:

- `cace` – Common Application Connection Establishment protocol instance processing `M_CONNECT` and `M_RELEASE` requests and responses;
- `auth` – providing authentication services during connection initialization); `cdap` (providing usual CDAP message exchange;
- `cdapSplitter` – delivering messages to appropriate upper submodules;
- `cdapMsgLog` – logger for an accounting of processed messages.

The `difAllocator` module handles locating a destination application based on its name. DA is a component of the DAP's IPC Management that takes ANI and access control information and returns a list of DIF-names through which the requested application is available. Moreover, the `difAllocator` module provides statically configured knowledge about simulation network graph. The `difAllocator` modules consists of five auxiliary submodules that maintain state information and help to deliver DA services:

- `da` – core functionality;
- `namingInformation` – mapping between APN synonyms;
- `directory` – mapping between APN and DIF-names;
- `searchTable` – mapping between APN and peer DA instance where to continue search;
- `neighborTable` – mapping between peer DA and neighboring DA instances.

The `ipcResourceManager` module currently queries DA module to find suitable IPCP and relays communication between AE and IPCP. The `ipcResourceManager` consists of two submodules:

- `irm` – acting as a broker between APs and IPCPs when handling the flow (de)allocation calls;
- `connectionTable` – maintaining state information for a given flows.

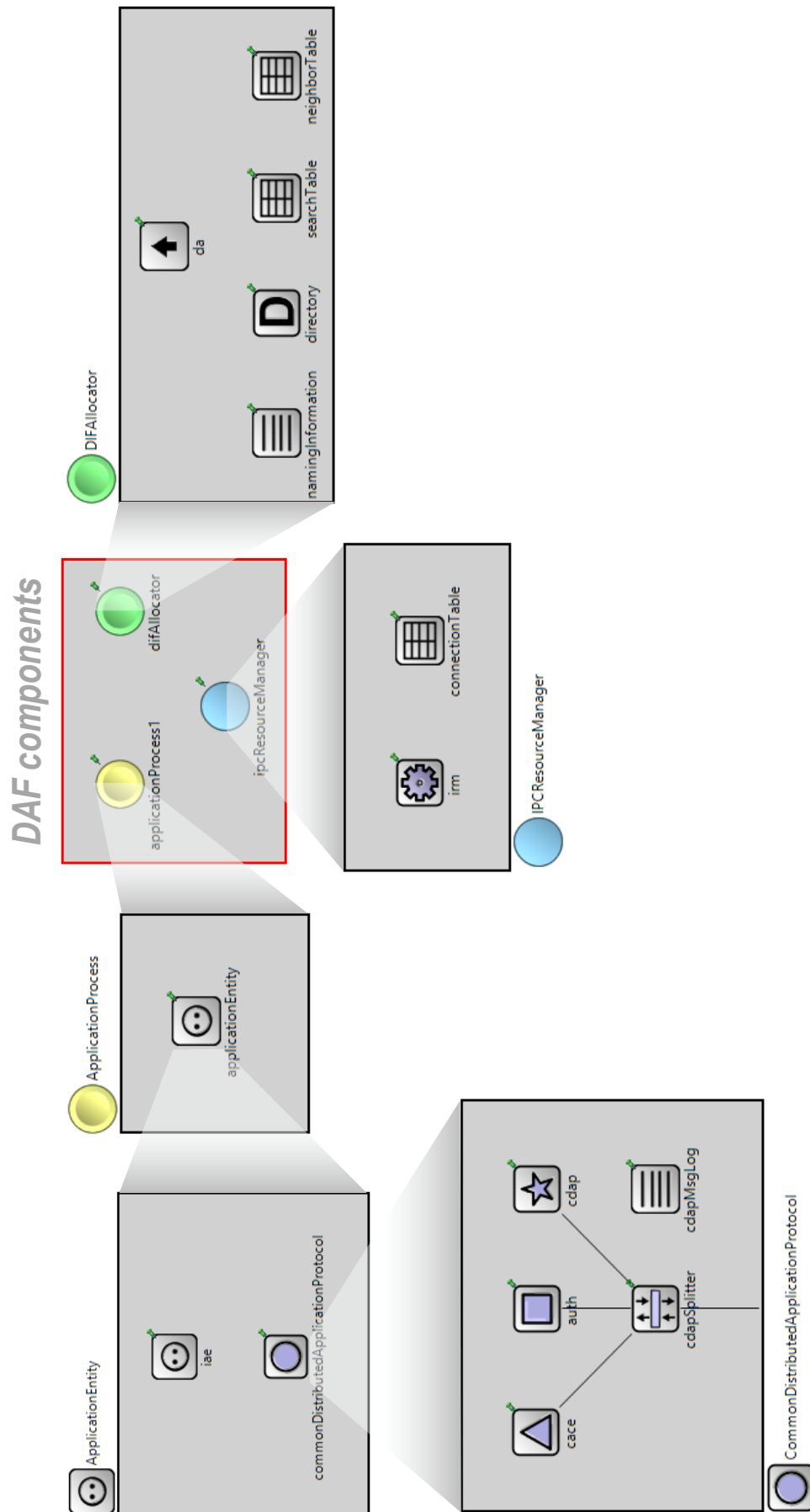


Fig. 67: DAF components for RINASim

DIF Modules

All currently implemented DIF components are enclosed to the `IPCProcess` container module (instantiation of `IPCP`). The `IPCProcess` contains following submodules, and overall structure is shown in Fig. 68:

- Enrollment, which governs enrollment of `IPCP` into DIF;
- Flow Allocator, which processes flow (de)allocation;
- EFCP, which provides data transfer services optionally with transfer control;
- Relaying and Multiplexing module, which handles incoming and outgoing PDUs;
- Resource Allocator, which monitors resources namely (N-1)-flows and available QoS;
- RIBDaemon, which is in charge of processing management messages;
- Routing policy, which maintains PDU forwarding rules.

The `enrollment` module is in charge of enrollment procedure, which occurs upon successful connection establishment between `IPCPs`. It consists of core functionality submodule and table (`enrollmentTable`) maintaining connection state of each enrollment FSM.

The `flowAllocator` module handles (de)allocation request and response calls from the `IRM`, `RIBDaemon` or `AE`. The `flowAllocator` module consists of three submodules (and currently three supported policy interfaces):

- `fa` – core functionality involving instantiation of FAIs;
- `nFlowTable` – mapping between (N)-flow and bound FAI;
- `fai_<portId>_<CEPid>` – managing a whole flow lifecycle.

The Error and Flow Control Protocol is modeled as one compound module. This module dynamically spawns `efcpi_<CEPid>` (EFCP instance) and `delimiting` submodules per one flow. There is also the `efcpTable` module maintaining bindings between `Delimiting` and `EFCPI`. Apart from that, the `MockEFCPI` processes management PDUs sent/received by local `RIBDaemon`. Each `EFCPI` contains the `ntp` submodule (providing data transfer services), the `ntpState` submodule (maintaining state-vector) and a few policies related to DTP functionality. Optionally, `EFCPI` may also contain the `dtcp` submodule and several DTCP policies, whenever transfer control is requested for a communication (i.e., due to the reliable transmission demand).

The `relayAndMux` module represents a stateless function that takes incoming PDUs and relay them within current `IPC` or pass them to an outgoing port. In particular the `RMT` takes PDUs from (N-1)-ports, consults their address fields and perform one of the following actions: a) relay PDU between (N-1)-ports; b) pass PDU to `EFCPI`; and c) multiplex PDU from `EFCPI` to (N-1)-port.

The `relayAndMux` consists of multiple simple modules of various types, some of them are static, and some of them are instantiated dynamically at runtime. Among dynamically created modules

are RMT ports (representing (N-1)-flow communication endpoints) and associated input/output queues. Among static submodules are:

- `rmt` – core functionality;
- `allocator` – managing addition, removal and reconfiguration of RMT queues and ports;
- `pduForwardingPolicy` – mapping table of destination addresses and QoS-ids to output ports that is used by the relaying functionality of the RMT;
- other policy module interfaces monitoring queue lengths and scheduling PDU departures.

The `resourceAllocator` monitors the operation of the IPCP and makes adjustments to its operation to keep it within the specified operational range. Its forwarding and queuing functionality are customizable by policies. The `resourceAllocator` consists of multiple simple modules of various types, namely:

- `ra` – core functionality that manages connections to other local IPCPs with the help of `nm1FlowTable` submodule;
- `pduFwdGenerator` – uses custom policies to manage `pduForwardingPolicy` entries;
- other policies executed upon RMT queue allocation.

The `ribDaemon` is the IPCP's management heart. It receives/sends CDAP management messages and notifies other submodules about management changes. RINASim's RIBDaemon consists of three submodules:

- `ribd` – core functionality mainly listening to calls from other DIF components and notifying them upon CDAP message reception;
- `commonDistributedApplicationProtocol` – same submodule as in case of DAF components description;
- `ribdSplitter` – splitter is delegating CDAP management messages to/from the `mockEFCPI` or appropriate EFCPIs.

The `routingPolicy` module is used by `pduFwdGenerator` to populate/update correctly the `pduForwardingPolicy`.

5.4.3 RINASim Demonstration

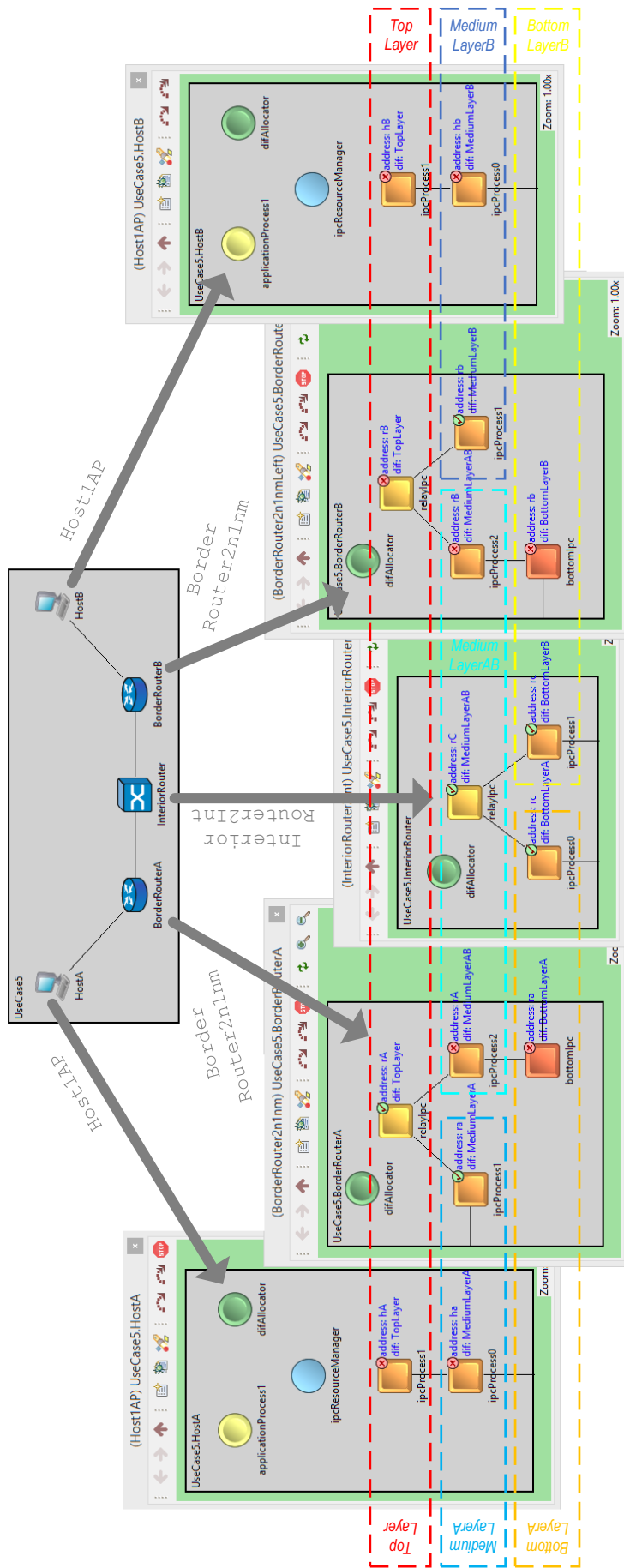
This subchapter presents one of the many demonstration RINA simulations available in RINASim. The goals are: a) to give a reader overview of RINASim capabilities; and b) to familiarize the reader with RINA concepts on simple computer network example.

Network Graph and Setup

The motivation behind this particular simulation is to show ping-like application communication within the simple network consisting of all different node types. Topology contains two host nodes (called *HostA* and *HostB*), two border routers (called *BorderRouterA* and *BorderRouterB*) and one interior router (called *InteriorRouter*) interconnected together as depicted in Fig. 69. Links between nodes are configured with one millisecond fixed transmission delay, which means that sending a packet from *HostA* to *HostB* takes four milliseconds.

There are totally six DIFs of three different ranks (network is just as in Fig. 52). Please notice addressing scheme where the same node may use the same address on different DIF as long as they are unambiguous within the layer's scope. RINA address length and syntax is policy-dependent. The demonstration uses flat address space with simple string addresses.

- Top most *TopLayer* DIF common to *HostA* (with address **hA**), *BorderRouterA* (address **rA** and self-enrolled), *BorderRouterB* (address **rB**) and *HostB* (**hB**);
- Three middle DIFs *MediumLayerA*, *MediumLayerAB* and *MediumLayerB*. *MediumLayerA* is common to *HostA* (**ha**) and *BorderRouterA* (address **ra** and self-enrolled). *MediumLayerAB* is common to *BorderRouterA* (**rA**), *InteriorRouter* (address **rC** and self-enrolled) and *BorderRouterB* (**rB**). *MediumLayerB* is common to *BorderRouterB* (address **rb** and self-enrolled) and *HostB* (**hb**).
- Two bottom most DIFs *BottomLayerA* and *BottomLayerB*. *BottomLayerA* is common to *BorderRouterA* (**ra**) and *InteriorRouter* (address **rc** and self-enrolled). *BottomLayerB* is common to *InteriorRouter* (address **rc** and self-enrolled) and *BorderRouterB* (**rb**).



All nodes topology with three levels of DIF

Fig. 69: RINASim demonstration topology

Multiple noticeable events happen during demonstration:

- 1) If another IPCP wants to communicate within a given DIF, then, it needs to be enrolled by a DIF member. Self-enrolled IPCPs are members of certain DIFs from the beginning of the simulation, and they help other IPCPs to join a DIF. In order to allow IPC between any node, the simulation is scheduled to commence enrollment of: *BorderRouterA* into *BottomLayerA* at $t=1s$; *BorderRouterA* into *MediumLayerAB* at $t=1.5s$; *BorderRouterB* into *TopLayer* at $t=2s$; and *HostB* into *TopLayer* at $t=5s$. The enrollment usually involves recursive calls of enrollment procedures in lower rank DIFs.
- 2) The IPC comprises of flow allocation, data transfer, and optional flow deallocation. *HostA* and *HostB* are configured for IPC using ping-like application (measuring one-way and round-trip delays). In this case, flow allocation is initiated at $t=10s$, first ping is sent at $t=15s$ and flow deallocation occurs at $t=20s$.

Source codes of demonstration are located in `/examples/Demos/UseCase5` folder and include following files, which may be used as templates when creating other RINASim scenarios:

- `UseCase5.ned` – OMNeT++ simulation network graph description which contains nodes and interconnections definitions;
- `omnetpp.ini` (see Addendum 8.6.1 for details) – scheduled simulation setup with models configuration (e.g., nodes addresses, ANI for AEs, pointers to XML configurations) applied during network initialization;
- `config.xml` (see Addendum 8.6.2 for details) – additional more structured and complex models configuration (e.g., DA’s mappings, RA’s QoS-cubes sets, preallocation and preenrollment settings) in the form of XML data is loaded to the simulation using this file;
- `*.anf` – statistic collection setup file(s);
- `./results/*` – results of various simulation runs containing gathered scalar and vector data.

By default, every RA contains implicit QoS Cube (with QoS-id “MGMT-QoSCube”) that defines QoS parameters (e.g., reliability, minimum bandwidth) for management traffic and guarantees successful mapping of management SDUs onto appropriate (N)-flow. Apart from this default QoS-cube, each RA loads QoS-cube set according to the simulation configuration. For demonstration, there are two more QoS-cubes available for each RA called “QoSCube-RELIABLE” and “QoSCube-UNRELIABLE” (same QoS parameters differing only in data transfer reliability). Please see Fig. 70 for visualization of loaded QoS-cube.

DA implementation currently allows only static change of its settings (namely different kinds of mappings). Hence, necessary configuration step is to initialize DA properly in order to provide services to FA, RA and other components depending on naming information. Namely two DA’s tables

are important for overall functionality – `Directory` (helps to search target IPCP for a given APN) and `NeighborTable` (used by FA to find a neighbor IPCP for a given IPCP). Fig. 71 shows shared directory information by all DA instances within the demonstration.

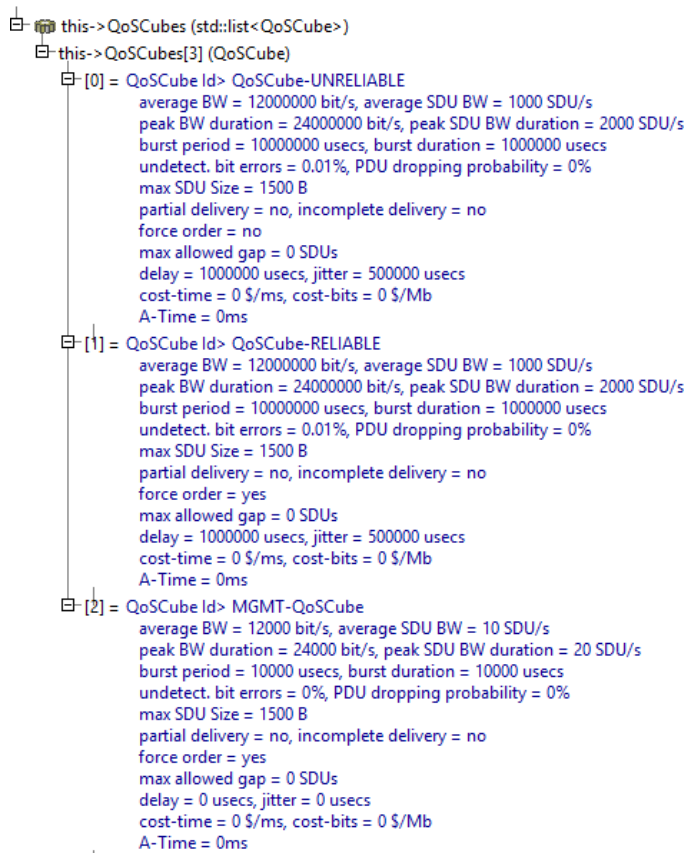


Fig. 70: Visualization RA's available QoS-cubes

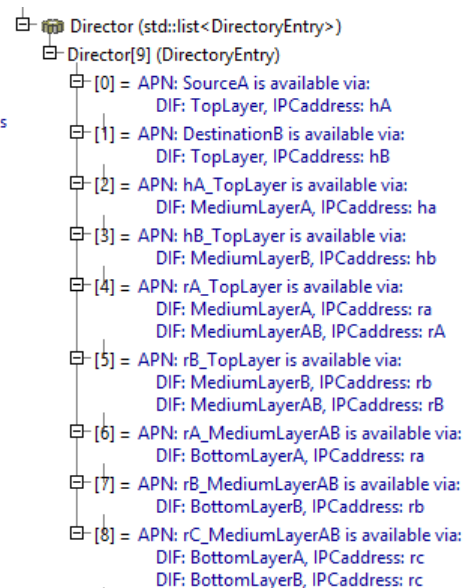


Fig. 71: Visualization of Directory mappings

Simulation description is divided into two subsections. All events connected with enrollment procedures are described in “Enrollment Phase” subsection and events related to data transfer between *HostA* and *HostB* are in “Data Transfer Phase” subsection. The most important parts are descriptions of the trivial enrollment use-case (steps marked with E*), trivial flow allocation use-case (steps marked with c*), trivial recursion call (steps marked with H*). They outline steps, which repeat upon similar use-cases employing recursive calls.

Enrollment Phase

Whole enrollment phase is divided into four events. The first event is enrollment of *BorderRouterA* into *BottomLayerA* at $t=1s$ with the help of *InteriorRouter* as enroller:

- E1) *ipcProcess2*'s Enrollment module of *BorderRouterA* is scheduled to join the DIF *BottomLayerA* just a second after the simulation started. Enrollment asks FA to provide management (N-1)-flow (with destination address **rc** of *InteriorRouter*) to carry CACEP messages. Because *bottomIpc* is 0-level DIF (i.e., it is directly connected to the medium), then RA returns

automatically successful binding of the (N-1)-flow – recursion cannot continue below 0-level DIFs;

- ε2) *ipcProcess0*'s Enrollment sends *M_CONNECT* (with *ra* as source and *rc* as the destination address) via RIBd to *InteriorRouter*. *ipcProcess0*'s Enrollment module leverages IPCP with address *rc* within *BottomLayerA* (which is *bottomIpc* of *InteriorRouter*) when joining this DIF. Because management (N-1)-flow is inherently present, management messages can be sent immediately. *M_CONNECT* opens application connection for management messages between *BorderRouterA*'s *bottomIpc* and *InteriorRouter*'s *ipcProcess0*;
- ε3) *bottomIpc*'s Enrollment replies with positive *M_CONNECT_R*. With this message (sent from *rc* to *ra*), *bottomIpc* of *InteriorRouter* accepts application connection;
- ε4) *ipcProcess0*'s Enrollment begins enrollment procedure by sending *M_START*.
- ε5) *InteriorRouter* responds with *M_START_R*. Both of these messages contains *EnrollmentObj* as abstract data structure holding relevant parameters such as current address, address expiration time and APN. *EnrollmentObj* allows to assign a dynamic address to newcomer DIF member. Nevertheless, this scenario works only with statically preconfigured addresses;
- ε6) Optionally, either *InteriorRouter* may send some *M_CREATE* messages to populate *BorderRouterA* RIB with information about neighboring IPCPs and their addresses. Alternatively, *BorderRouterA* may ask for this information using *M_READ* messages. Alternatively, alternatively, both can exchange some authentication objects proving the identity of communicating parties.
- ε7) However, let us consider the simplest case, where *bottomIpc*'s Enrollment sends *M_STOP* immediately after *M_START_R*. *InteriorRouter* ends enrollment procedure because it has all the necessary information from a joining member;
- ε8) *ipcProcess0*'s Enrollment replies with *M_STOP_R*. *BorderRouterA* finalizes enrollment by sending this message as Acknowledgement.

The previous description outlines the most straightforward enrollment procedure that happens between joining member and enroller. The contents of *EnrollmentStateTable* (as abstract data structures holding information for IPCP's DIF membership) illustrating above-mentioned event is available in Addendum 8.6.3. Subsequent descriptions mention only notable changes because enrollment steps ε1-ε8 (CACEP message exchange) are present in all of them.

The second event is joining of *BorderRouterA* into *MediumLayerAB* at $t=1.5s$ once again with the help of *InteriorRouter* as enroller:

- c1) *BorderRouterA*'s *ipcProcess2* is scheduled with enrollment procedure to join *MediumLayerAB* leveraging *InteriorRouter*. Both IPCPs needs communication channel through which they may

exchange management messages. Hence, *BorderRouterA*'s FA of *ipcProcess2* receives request for management flow (from Enrollment module) and asks RA to allocate appropriate (N-1)-flow (with source *ra* and destination *rc*) for communication with *InteriorRouter*'s IPCP with address *rC*;

- c2) *ipcProcess2*'s RA bothers *bottomIpc*'s FA with allocation request because destination name resolution returned *bottomIpc* IPCP as being in the same DIF as IPCP with address *rc*. *bottomIpc*'s FA creates EFCPI to handle this data transfer (from perspective of *bottomIpc* this communication is just another data flow);
- c3) *bottomIpc*'s FA sends *M_CREATE* containing *Flow* object inside via RIBd (because *bottomIpc* is already enrolled to the DIF *BottomLayerA*). *Flow* object describes all properties including source's and destination's addresses, port-ids, CEP-ids, QoS demands and chosen QoS Cube (in case of management messages it is always predefined QoS Cube with id "MGMT-QoS Cube");
- c4) *M_CREATE* is delivered to *ipcProcess0*'s RIBd and FA, where it initiates the procedure for processing of *create request flow*. On *InteriorRouter*, *ipcProcess0* IPCP represents (N-1)-DIF for flow and *relayIpc* IPCP represents (N)-DIF for connection. Hence, *ipcProcess0*'s FA notifies *relayIpc* about possible flow allocation. *relayIpc*'s RIBd delegates this call to RA and Resource Allocator decides whether it has enough resources to accept or not the new flow.
- c5) *ipcProcess0*'s FA replies with positive *M_CREATE_R*. *relayIpc*'s RA responded positively to allocation call. Therefore, *ipcProcess0*'s FA instantiates opposite EFCPI, which involves the assignment of local port-id/CEP-id and bindings of gates. Following this, *ipcProcess0*'s FA asks *ipcProcess0*'s RIBd to generate and dispatch *M_CREATE_R* with updated *Flow* object stating successful flow allocation;
- c6) *bottomIpc*'s FA receives *M_CREATE_R* and notifies *ipcProcess2*'s RA about it. FA updates local *Flow* object. Flow is effectively in place as a channel for communication between *BorderRouterA*'s *ipcProcess2* and *InteriorRouter*'s *relayIpc*. Hence, RA is alerted about (N-1)-flow being ready and handles control back to Enrollment module;
- c7) Subsequently, steps e1-e8 repeats, where IPCP with address *rA* (*BorderRouterA*) is enrolled into *MediumLayerAB* by IPCP with address *rC* (*InteriorRouter*).

Create request/response flow calls are always accompanied by aforementioned steps c3-c7 and exchange of *M_CREATE* and *M_CREATE_R* messages. State information for each flow are stored in *flowAllocator*'s submodule called *nFlowTable*. Illustration of related *BottomLayerA*'s state tables is depicted in Fig. 84.

The third event is an enrollment of *BorderRouterB* into *TopLayer* at $t=2s$. Enrollment is scheduled on the top ranked IPCP (which is *relayIpc*) using *BorderRouterA* as enroller. Nevertheless, neither *BorderRouterB*'s *ipcProcess2*, nor *BorderRouterB*'s *bottomIpc* is enrolled to its DIF. Hence,

MediumLayerAB enrollment must occur before *TopLayer* enrollment, and *BottomLayerB* enrollment must precede *MediumLayerAB* enrollment:

- #1) *relayIpc*'s Enrollment asks FA for management (N-1)-flow in order to send CACEP messages from **rB** to **rA** within *TopLayer*. Because it does not exist, RA delegates flow allocation to *ipcProcess2*;
- #2) *ipcProcess2*'s FA receives a call. FA checks whether there is management (N-1)-flow for *create request flow* messages between **rB** (*BorderRouterB*'s *ipcProcess2*) and **rA** (*BorderRouterA*'s *ipcProcess2*) within *MediumLayerAB*. There is none flow and more over *BorderRouterB*'s *ipcProcess2* is not even enrolled into *MediumLayerAB*. Hence, *ipcProcess2*'s FA notifies RA that it need underlying management (N-1)-flow (from perspective of *relayIpc* it is (N-2)-flow) for enrollment procedure;
- #3) *bottomIpc*'s FA receives a call. Because *bottomIpc* is in 0-level DIF, then RA returns automatically successful binding of the management (N-1)-flow. Enrollment procedure occurs between *BorderRouterB*'s *bottomIpc* and *InteriorRouter*'s *ipcProcess1*, which both are in *BottomLayerB* DIF. Basically, IPCP with address **rb** successfully enrolls into *BottomLayerB* using IPCP with address **rc** going through steps e1-e8;
- #4) *bottomIpc*'s FA is notified about successful enrollment into *BottomLayerB* and continues with flow allocation initiated during step #3. Hence, *BorderRouterB*'s *bottomIpc* and *InteriorRouter*'s *ipcProcess1* RIBds and FAs exchange messages as in steps c3-c7. Eventually, management flow between **rb** and **rc** for *MediumLayerAB* communication is ready, and *BorderRouterB*'s *ipcProcess2* is alerted about this;
- #5) *ipcProcess2*'s RA is notified about successful management flow allocation. Hence, enrollment procedure initiated in step #2 may continue. IPCP with address **rB** (*ipcProcess2* of *BorderRouterB*) successfully enrolls into *MediumLayerAB* using IPCP with address **rC** (*relayIpc* of *InteriorRouter*) going through steps e1-e8;
- #6) *ipcProcess2*'s FA is notified about successful enrollment into *MediumLayerAB* and continues with flow allocation initiated during step #2. Hence, *BorderRouterB*'s *ipcProcess2* and *BorderRouterA*'s *ipcProcess2* exchange *create request/response flow* as in steps c3-c7. Notable difference comparing to flow allocation in step #4 is that messages pass through *InteriorRouter* (namely its *relayIpc*) as an interim device. Management flow between *InteriorRouter*'s *relayIpc* and *BorderRouterA*'s *ipcProcess2* is already present as the result of the second event of "Enrollment Phase". Eventually, management flow between **rC** and **rA** for *TopLayer* communication is in place, and *BorderRouterB*'s *relayIpc* is informed;
- #7) *relayIpc*'s RA is notified about successful management flow allocation. Hence, enrollment procedure initiated in step #1 may continue. All underlying connections are ready, and data path for management messages exists between *BorderRouterB* and *BorderRouterA* on relevant DIFs.

IPCP with address **rB** (*relayIpc* of *BorderRouterB*) successfully enrolls into *TopLayer* using IPCP with address **rA** (*relayIpc* of *BorderRouterA*) going through steps E1-E8.

The fourth and the last event is an enrollment of *HostB* into *TopLayer* at $t=5s$. Enrollment is scheduled on the top ranked IPCP (which is *ipcProcess1*) using *BorderRouterB* as enroller. Nevertheless, *BorderRouterB*'s *ipcProcess0* is also not enrolled into its DIF (*MediumLayerB*). Hence, *MediumLayerB* enrollment must occur before *TopLayer* enrollment. Situation is similar due to the recursions as in previous use-cases. Hence, we will omit unnecessary details when describing this event:

- н1) *HostB*'s *ipcProcess1* checks existence of management (N-1) flow between *HostB*'s *ipcProcess0* and *BorderRouterB*'s *ipcProcess1*. There is none flow. Thus one must be allocated before enrollment procedure on *TopLayer*;
- н2) Flow allocation call descend to *HostB*'s *ipcProcess0*. Over there is also as the first thing checked whether management (N-1) flow exists. Because *ipcProcess0* is in 0-level DIF, binding of (N-1)-flow is automatically successful;
- н3) *HostB*'s *ipcProcess0* (with address **hb**) enrolls into *MediumLayerB* DIF using *BorderRouterB*'s *ipcProcess1* (with address **rb**) as enroller going through steps E1-E8;
- н4) After *HostB* is successfully enrolled into *MediumLayerB*, management flow allocation from step #2 continues. The flow between *HostB*'s *ipcProcess0* and *BorderRouterB*'s *ipcProcess1* is created employing steps c3-c7. This flow is going to carry as data CACEP signalization messages between *HostB*'s *ipcProcess1* and *BorderRouterB*'s *relayIpc*;
- н5) *HostB*'s *ipcProcess1* is notified about management (N-1) flow presence and enrollment procedure initiated in #1 continues. *HostB*'s *ipcProcess0* (with address **hb**) is enrolled into *TopLayer* DIF leveraging *BorderRouterB*'s *relayIpc* (with address **rB**).

The final state after “Enrollment Phase” is that all nodes IPCPs are enrolled (or self-enrolled) into their DIFs except *HostA*'s IPCPs. All flows created during “Enrollment Phase” carries only CACEP messages (for connection establishment) and they are intended for direct RIBd-to-RIBd communication employing various management messages, thus, these flows are called **management flows**.

Data Transfer Phase

The main outcome of this scenario is a simulation of data transfer events between *HostA* and *HostB* employing ping-like application (AEMyPing). This application sends probe request (*M_READ*) from *HostA* to *HostB*, where *HostB* replies with the response (*M_READ_R*). One-way and round-trip time delays are measured employing this simple application.

“Data Transfer Phase” is divided into three notable events – flow allocation, data transfer, and flow deallocation. We will describe them in a similar fashion as the previous phase.

Data flow allocation starts at $t=10s$. *HostA*'s *applicationProcess1* (with APN *SourceA*, API-id 0, AEN *MyPing*, AE-id 0 as ANI parameters) requests flow for communication with *HostB*'s *applicationProcess1* (with APN *DestinationB*, API-id 0, AEN *MyPing*, AE-id 0 as ANI parameters). Event goes through following set of steps:

- #1) *Allocate request* is delivered to IRM. Over there, DA is asked to resolve destination ANI onto IPC address within certain DIF available to *HostA*. The following result is returned yielding that *DestinationB* is reachable via IPCP **hB** in *TopLayer* DIF;
- #2) *HostA* can access *TopLayer* leveraging *ipcProcess1*. Hence, IRM delegates allocate request call to *ipcProcess1*'s FA. As usually, FA instantiates EFCPI and verifies whether IPCP is enrolled into DIF before any attempt for sending *create request flow* (analogous to steps c1-c2). The situation is now similar to enrollment procedure of *HostB* because neither *ipcProcess1* nor *ipcProcess0* are enrolled into their DIFs. Therefore, *HostA* repeats same steps #1-5, which involve following actions performed due to the recursive calls in this order of finalization: a) enrollment of *HostA*'s *ipcProcess0* into *MediumLayerA* by *BorderRouterA*; b) creation of management flow between IPCP **ha** and IPCP **ra** within *MediumLayerA*; c) enrollment of *HostA*'s *ipcProcess1* into *TopLayer* by *BorderRouterA*;
- #3) After successful enrollment of *ipcProcess1*, FA may continue with flow allocation. FA exchanges *create request/respond flow* with *HostB* (analogously to c3-c7). This includes the creation of (N-1)-flow between **ha** and **ra** in *MediumLayerA* and creation of (N)-flow between **hA** and **hB** in *TopLayer*. However, it gets more complex in *TopLayer* DIF because *M_CREATE* and *M_CREATE_R* messages must be relayed by border routers to reach *HostB*, which causes additional recursive flow allocations between interim devices (i.e., *BorderRouterA*, *InteriorRouter*, *BorderRouterB*). All interim devices are already enrolled into their DIFs, thus established flows serve as carriers for *HostA* and *HostB* data transfer. The next steps briefly describe this multi-action step;
- #4) *M_CREATE* from *HostA* to *HostB* is received by *BorderRouterA*'s *relayIpc*. *BorderRouterA* inspects *create request flow* and determines *BorderRouterB* with the help of DA as the next-hop. Because border routers are not directly connected, they can communicate via *InteriorRouter* as a proxy. Therefore, *BorderRouterA* establishes flow between **ra** and **rc** of *BottomLayerA* and sends *create request flow* in *MediumLayerAB*.
- #5) *M_CREATE* from *BorderRouterA* to *BorderRouterB* is received by *InteriorRouter*'s *relayIpc*. The message needs to be relayed to *BorderRouterB*. Hence, flow is created between **rc** and **rb** in *BottomLayerB*. Then, *create request flow* is forwarded within this DIF;
- #6) *M_CREATE* from *BorderRouterA* to *BorderRouterB* within *MediumLayerAB* is received by *BorderRouterB*'s *ipcProcess2*. *BorderRouterB* accepts flow and sends *create respond flow* that

travels back to *BorderRouterA*. Because flow connecting both border routers (*rA* and *rB* within *MediumLayerAB*) is established, flow allocation from #4 may continue;

- #7) *M_CREATE* from *HostA* to *HostB* is received by *BorderRouterB*'s *relayIpc* after passing through flows created during #5 and #6. *BorderRouterB* inspects *create request flow* and determines that *HostB* is reachable via its *MediumLayerB*. In order to successfully relay *M_CREATE* to its final destination, *BorderRouterB* allocates flow between *rb* and *hb* in *MediumLayerB*. Subsequently, *M_CREATE* is forwarded to *HostB*;
- #8) *M_CREATE* is received by *HostB*'s *ipcProcess1*. FA notifies *applicationProcess1* about current flow allocation. *applicationProcess1* accepts flow for data transfer between APs. The decision is returned to *ipcProcess1*'s FA. IRM is asked to create bindings between AP and IPCP. FA instantiates EFCPI, updates *Flow* object and replies back to requestor with *M_CREATE_R*;
- #9) *M_CREATE_R* is relayed via all flows formed during #4-#7 to *HostA* until *ipcProcess1*'s FA receives this message. FA updates *Flow* object and notifies *applicationProcess1* about successful flow allocation. Then IRM adds missing bindings and whole data path between *HostA* and *HostB* is ready. (N)-flow in *TopLayer* can carry data traffic between AEs with the help of all underlying flows.

The next event is a transfer of data traffic between AEs. *HostA* sends five ping-like probes employing individual object inside *M_READ* message starting at $t=15s$. Upon reception of these messages, *HostB* replies with probe response, which is dedicated *M_READ_R* message. Data path and consistent flows are depicted in with different colors to get oriented in the following the description. Event consists of five repetitions of two steps:

- #1) *HostA*'s *applicationProcess1* sends a *M_READ* message, which is passed through IRM into *ipcProcess1* to *flow* prepared during the previous event and descends to *ipcProcess0*. The message travels through the medium and *flow* connecting *HostA* with *BorderRouterA* within *MediumLayerA*, where it is received by *ipcProcess1*. It is relayed by *BorderRouterA*'s *relayIpc* to *ipcProcess2* and *flow* interconnecting *BorderRouterA* and *BorderRouterB* in *MediumLayerAB*. Because border routers are not directly connected, the message is passed to a lower *bottomIpc* into *flow* interconnecting *BorderRouterA* with the neighboring *InteriorRouter* in *BottomLayerA*. Message traverses through the medium and it reaches *InteriorRouter*'s *ipcProcess0*. Over there, message ascends to *relayIpc*, where is relayed within *MediumLayerAB*. Then it descends to *ipcProcess1* into *flow* interconnecting *InteriorRouter* and *BorderRouterB* in *BottomLayerB*. The message travels through medium to *BorderRouterB*'s *bottomIpc*. It ascends to *ipcProcess2* and is relayed by *relayIpc* to *ipcProcess1*. Finally, the message reaches *HostB*'s *ipcProcess0* through medium inside *flow* within *MediumLayerB*. It ascends to *flow* in *ipcProcess1* (member of *TopLayerB*) and through IRM to *HostB*'s *applicationProcess1* as recipient;

#2) *HostB's applicationProcess1* responds with *M_READ_R* message that returns to *HostA* traveling in opposite direction through the same data (marked with violet line) path as in #1. Depending on direction message is either encapsulated (from *HostA* to *HostB* green circles) or decapsulated (from *HostA* to *HostB* orange circles) into/from PDU or relayed (brown circles).

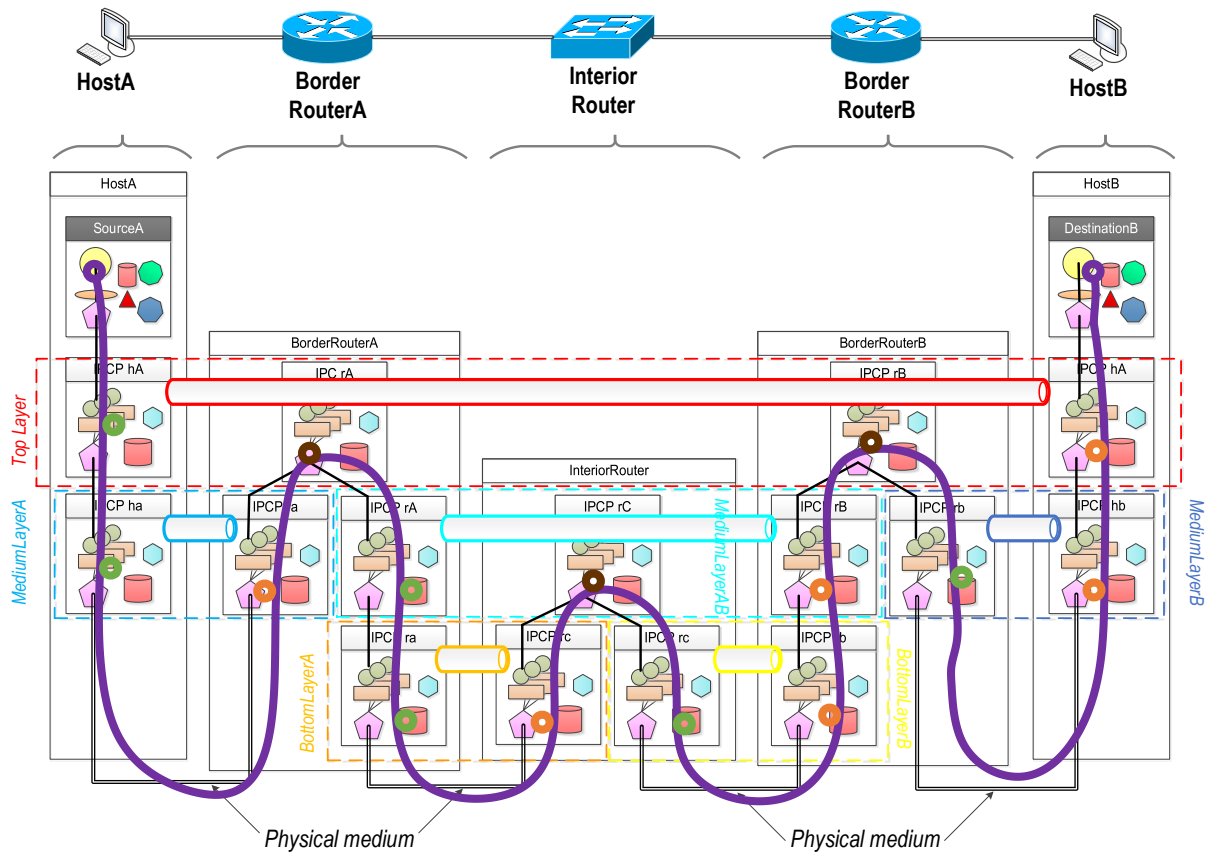


Fig. 72: Data transfer phase illustration

After APs exchanged pings, *HostA's* AE closes the connection and sends *deallocate submit* to *HostB* at $t=20s$. Deallocation affects only flow present in *TopLayer*. Current RINASim implementation leaves underlying (N-1/2)-flows (i.e., those not directly connected with APs) intact because they may be reused later by other applications. This event is accompanied by following steps:

- #1) *HostA's applicationProcess1* tells IRM to deliver *deallocate submit*. IRM disconnects from its side port binding. Then, IRM delegates flow deallocation to *ipcProcess1's* FA;
- #2) This FA generates a *M_DELETE* message with updated *Flow* object state inside and sends it towards *HostB* through *flow* in *TopLayer*. Message follows data path leveraging existing management flows created during enrollment phase;
- #3) *HostB's ipcProcess1* receives *M_DELETE*. FA updates its version of *Flow* object. FA delivers *deallocation submit* to *HostB's applicationProcess1*, which tells IRM to remove bindings.
- #4) *ipcProcess1's* FA on *HostB* then replies with *M_DELETE_R* acknowledging successful flow deallocation. This message is carried back to *HostA*;

#5) *HostA's ipcProcess1* receives *M_DELETE_R*. FA marks flow as deallocated and disconnects remaining bindings between IPCP and IRM.

The result of flow (de)allocation and flow's state is maintained in *ipcProcess1's* *NFlowTable* of *HostA* and *HostB*. We can inspect flow parameters in these tables as illustrated in Fig. 73. We can see that two EFCPIs handled endpoints of data transfer – EFCPI with CEP-id 18 430 in *HostA's ipcProcess1* and EFCPI with CEP-id 60 067 in *HostB's ipcProcess1*. Bindings between AP and IPCP are ports identified with port-id 7 877 for *HostA* and port-id 57 495 for *HostB*. The only QoS demand by *AE MyPing* is the reliability of data transfer (expressed with QoS attribute “force order” set to true). Therefore, RA assigned QoS Cube named “QoSCube-RELIABLE” to flows requested by this AE. Flow object between *HostA* and *HostB* in *TopLayer* was created at $t=10s/10.026s$ and was deleted at $t=20.008s/20.004s$.

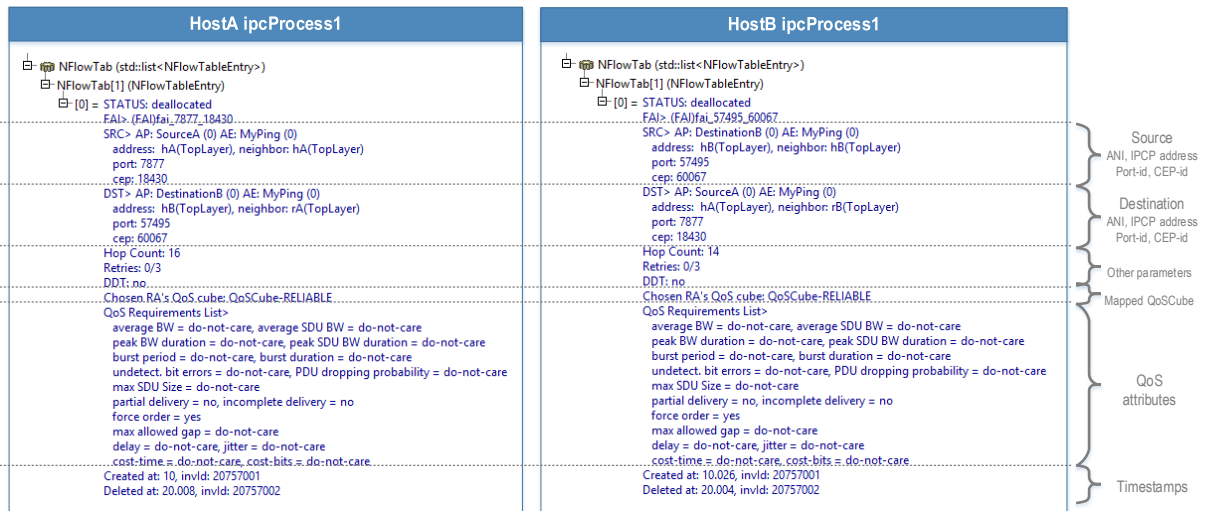


Fig. 73: Content of TopLayer ipcProcess1 NFlowTables for HostA and HostB

5.5 Chapter Summary

In this chapter, we described core RINA principles. We tried to summarize RINA theory in the text that lacks any usage of the term without previous thorough definition / context explanation because we know, how hard the “mental shift” from TCP/IP concepts towards RINA is.

The second subchapter went into more details about various RINA components. It started with a description of different kinds of high-level RINA nodes including hosts, interior routers, and border routers. Subsequently, we dived deep into low-level RINA components that are being used by DIF and DAF. Besides that as the research contribution, we thoroughly analyzed and enhanced (particularizing functional descriptions and equipping them with FSMs) RINA specifications namely for FA and CACEP operation.

Subchapter 5.3 briefly mentioned relevant research grant activities and available implementations to reveal current RINA state-of-the-art.

The last subchapter described RINASim including installation guideline, design notes, and demonstration. RINASim philosophy benefits from clever OMNeT++ module interfacing in order to allow flexible change of used policies. Moreover, Subchapter 5.4 ending contained a detailed illustration of RINA principles using RINASim demo scenario. Demonstration description should show the impact of recursion and help others to understand enrollment and flow (de)allocation procedures in praxis. Moreover, demonstration setup may be employed as the template when creating new scenarios.

We have designed and implemented RINASim as the first full-scale RINA simulator containing a wide gamut of functions that are extensible and replaceable. RINASim reliably proves following RINA properties: isolation of *namespaces* and address spaces across DIFs; enrollment and flow allocation recursion and their impact; routing based on available resources reflecting QoS attributes; easy application protocol prototyping when employing CDAP messages (and action primitives they substitute); and others. Hence, RINA offers by design complete naming scheme and fulfills most of the ideal solution properties as described in Chapter 3.

The main contribution of this chapter is RINASim as a tool that helps: 1) researchers to prototype and test new policies and mechanisms in native and full-compliant RINA environment; 2) others to visualize and understand RINA principles.

6 Conclusion

☞ –“A story has no beginning or end: arbitrarily one chooses that moment of experience from which to look back or from which to look ahead.” Graham Green

☞ *What has been done and accomplished in frame of this dissertation thesis?*

☞ *What are the important results?*

We pursue a difficult and complex task to define and to discuss elementary naming, addressing and routing principles of computer networks.

The thesis begins with an overview of networking fundamentals and points out design issues of traditional TCP/IP stack that are becoming more apparent as more users and devices are accessing the Internet each day. We tried to qualify causes and quantify their (future) impact (when following current trends). Internet developed incrementally throughout previous 40 years. However, the Internet struggles to redesign its communication schemes after the adoption of TCP/IP and its global expansion.

We collected and studied relevant papers and works written on the topic of naming, addressing and routing. We formulated low-level foundations using formal math apparatus. We compiled encompassing high-level theory and checked its compliance among existing addressing and naming techniques. This work allowed us to reevaluate problems of current Internet in the new light, which confirmed that aforementioned problems of TCP/IP are consequences of incomplete architecture that lacks necessary levels of indirection. We investigated properties of existing candidates, which aspire to deal with this situation. We decided to follow LISP and RINA further with our research efforts.

We thoroughly analyzed LISP use-cases and protocol details (namely the split of locator address space and identifier *namespace*). We were able to identify and investigate certain shortcomings of LISP design. Based on that, we developed improvements to LISP operations and verified them using discrete event simulator. We implemented the first low-level LISP simulation modules and successfully checked their compliance with the referential Cisco implementation in the real network. The principle of our LISP research is included in papers [131], [132] and [133].

We conducted a similar analysis of RINA and its properties that aim to the clean-slate design of not only naming and addressing but also other aspects of computer networking. We revisited all available RINA specifications and try to improve their clarity, particularly parts describing enrollment and flow (de)allocation procedures. Subsequently, we designed and implemented the first RINA discrete event simulator called RINASim, which provides a standalone framework with full-fledged RINA simulation modules for OMNeT++. The core contribution of our RINA research has been published as an independent framework in [171] and explained in PRISTINE Deliverable 2.4 [167] and Deliverable 2.6 [172].

Following two subchapters outline some conclusions and results of our research efforts involving Locator/Id Separation Protocol and Recursive InterNetwork Architecture.

6.1 Summary about LISP

Precise LISP (and VRRP) simulation modules for OMNeT++, which are used as the basis for ongoing research, represent the main code contribution. Based on well-known designed issues (see [108]), we investigated, proposed, implemented and tested two improvements – map-cache synchronization and merged RLOC probing. Our map-cache synchronization techniques minimize map-cache misses, thus significantly decreasing packet loss. Furthermore, employing our merged RLOC probing algorithms has an outstanding impact on LISP protocol overhead comparing to simple RLOC probing per every EID.

Despite the accomplished achievements in LISP operation tuning, LISP is unfortunately not an ultimate solution for current Internet troubles. It breaks several RFC 1958 concepts, and some problems were revealed during its worldwide deployment (RFC 7215 [173]). Moreover, LISP deployment needs additional configuration effort to secure LISP against possible attacks and threats (see [174]).

Basically, any solution decoupling locator and identifier has to deal with Locator Path Liveness problem, and any non-host-based loc/id split has to cope with Site-based State Synchronization problem. Their impact can be diminished (with for instance map-cache synchronization described above) but not completely treated. Hence, neither LISP nor any CES/CEE proposal reviewed in Chapter 3.5 is the desired solution.

Another and probably the most serious rebuke of any hybrid or network-based loc/id split is when a packet is traversing locator *namespace* then the routing is performed according to the locator, not an identifier. Previous is strictly in contradiction to the theory reviewed in Chapter 3, and implications are thoroughly investigated in [126]. LISP suffers from three major problems:

- 1) Routing should be done based on *node names* (see Saltzer's [49]). However, "routes" in nowadays Internet use PoAs (IP addresses in Fig. 14). Therefore, all IP "routing" is based on false premises and would always be *route dependent* (which is unwanted based on knowledge in Subchapter 3.3). Routing should be performed based on identifier not locator (otherwise, it leads to Locator Path Liveness problem);
- 2) Locator and identifier are not *bound* to the same *object* – locator address is an address of the interim device (which performs header alternation relevant to loc/id split) not the end-device of communication;
- 3) All identifiers are used in some sense also for locating. An *object* cannot be *located* without *identifying* it and vice versa (see Saltzer's [45]). There could neither be identification without localization, nor localization without identification. Thus, there should be no semantic distinction between identifier and locator on the Internet but yet there is.

Therefore, LISP does not provide proper naming and addressing concept, nor it is even scalable routing solution for TCP/IP architecture.

6.2 Summary about RINA

RINA as the new (and complete) clean-slate architecture tries to touch and codify every part of communication within computer networks. Therefore, RINA's knowledge base spans from high-level reference model description to low-level characterization of each component functionality. Pouzin Society [175] is a formal body in charge of maintaining specifications with FIT-BUT as one of its members. In the theoretical part of this dissertation, we revisited and extended parts of RINA specifications concerning flow allocation and connection establishment procedure. We supplemented them with FSMs illustrating FA and CACE operations.

RINASim is the main contribution, and RINASim's development process helped to clarify and progress some RINA specifications. As the RINASim's chief designers and implementers, we authored FA, DA, AE, RIBd and RA simulation modules in the frame of this thesis.

RINA is still young in its technological readiness level. Hence, some of RINA's concepts were doubtful whether they will work or not. Following RINA features would not be possible to prove or verify without RINASim:

- We simulated and shown basic RINA functionality (enrollment, flow allocation and data transfer) in this thesis (and in [171]). RINA can achieve IPC employing recursively the same (DIF and DAF) components, which simplifies implementation of the network stack. Furthermore, DIF scope isolation allows reusing IPCP's APNs without any duplicity address problems. Hence, there is no need for global address space due to the DIF isolation;
- RINA allows an easy employment of **Aggregated Congestion Control (ACC)**, see PRISTINE Deliverable 3.2 [176] for more. ACC improves QoS experience for communicating parties whenever congestion occurs in the network. RINA offers built-in mechanisms with programmable policies to handle resource allocation in compliance with QoS demands;
- Custom routing algorithm taking into account division of (*location dependent*) *address space* reduces significantly routing table sizes for distributed cloud installations. Aforementioned solution – called **Scalable Forwarding with RINA (SFR)**, see paper [177] for details – provides proofs for real-life use-case that topologically dependent (hierarchical) addresses help in routing comparing to flat address space.

RINA theory seems to offer complete naming, addressing and routing concepts. Moreover, RINA's design separating mechanisms and policies is flexible enough to allow scalable changes reflecting demands of future Internet. Nevertheless, RINA needs more validation and verification testing (preferably) on real-life deployment to support previous claims.

6.3 Future Work

We take this thesis just as the beginning of more advanced research involving OMNeT++ simulator as a validation tool for new routing paradigms (such as LISP) and alternative architectures (such as RINA).

We would like to discuss our LISP improvements – map-cache synchronization and merged RLOC probing – within IETF to see whether they can be submitted as draft proposals. Our plans with LISP simulation modules include to add support for proxy xTR functionality and to recognize more LISP control flags (like SMR bits). We would like to use further our LISP simulation modules and test effectiveness of different distributed mapping systems (e.g., LISP-ALT, LISP-DDT). Also, we intend to upgrade VRRP to support IPv6 addresses and all features of VRRP version 3. We would like our low-level LISP simulation modules to be considered as the verification tool for other LISP related use-cases and technologies. Therefore, we want to integrate LISP source codes with official INET framework as the first step (which is something we already accomplished before [178] or [179]).

We plan to carry on work on RINA research topics and further refine RINASim based on new knowledge and up-to-date specifications. An additional goal is to conduct a comparative evaluation of our simulation models with RINA implementation for Linux environment called IRATI. Adoption of the newest version 6.5 of EFCP, SDU protection module integration, NSM and dynamic DA functionality are on our development roadmap for the nearest future.

6.4 Final Thoughts

This subchapter contains a few thoughts that helped to shape this thesis and that are worthy to be considered by any young scientist interested in computer networks.

— *“The Internet is at its core an unfinished demo.”*

John Day

— *“(6) It is easier to move a problem around (for example, by moving the problem to a different part of the overall network architecture) than it is to solve it.”*

— *“(11) Every old idea will be proposed again with a different name and a different presentation, regardless of whether it works.”*

RFC 1925 [180]

— *“Ninety percent of everything is crap!”*

Theodore Sturgeon

— *„Hlavu vzhůru nos mezi mraky!”*

Arnošt Veselý

7 Bibliography

- [1] ISO, "Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model". Patent ISO/IEC 7498-1:1994, 1994.
- [2] B. Carpenter, "RFC 1958: Architectural Principles of the Internet," June 1996. [Online]. Available: <http://tools.ietf.org/html/rfc1958>.
- [3] J. Day, *Patterns in Network Architecture: A Return to Fundamentals*, Boston: Prentice Hall, 2008.
- [4] P. Ji, Z. Ge, J. Kurose and D. Towsley, "A Comparison of Hard-State and Soft-State Signaling Protocols," *IEEE/ACM Transactions On Networking*, vol. 15, no. 2, pp. 281-294, 2007.
- [5] R. Watson, "Delta-t Protocol Specification," Lawrence Livermore Laboratory, December 1981. [Online]. Available: <http://www.osti.gov/scitech/servlets/purl/5542785>.
- [6] S. Floyd, "RFC 2914: Congestion Control Principles," September 2000. [Online]. Available: <https://tools.ietf.org/html/rfc2914>. [Accessed January 2016].
- [7] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd and V. Jacobson, "RFC 2309: Recommendations on Queue Management and Congestion Avoidance in the Internet," April 1998. [Online]. Available: <https://tools.ietf.org/html/rfc2309>. [Accessed January 2016].
- [8] M. Allman, V. Paxson and E. Blanton, "RFC 5681: TCP Congestion Control," September 2009. [Online]. Available: <https://tools.ietf.org/html/rfc5681>. [Accessed January 2016].
- [9] M. A. Rahman, A. Pakštás and F. Z. Wang, "Towards Communications Network Modelling Ontology for Designers and Researchers," in *Proceedings of International Conference on Intelligent Engineering Systems INES '06*, London, United Kingdom, 2006.
- [10] Cisco Systems, Inc., "The Zettabyte Era Trends and Analysis - Cisco," May 2015. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/VNI_Hyperconnectivity_WP.html. [Accessed September 2015].
- [11] R. Atkinson, "IPv6 Routing Table Size Issues," October 1996. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-ipngwg-ipv6-routing-00>. [Accessed January 2016].
- [12] J. Yu, "RFC 2791: Scalable Routing Design Principles," July 2000. [Online]. Available: <https://tools.ietf.org/html/rfc2791>. [Accessed January 2016].
- [13] M. Grégr and T. Podermanski, "IPv6 @ CESNET network | 6lab.cz," Brno University of Technology, September 2015. [Online]. Available: <http://6lab.cz/live-statistics/ipv6-cesnet-network/>. [Accessed September 2015].

- [14] IRTF, "Routing Research Group (RRG)," [Online]. Available: <https://irtf.org/concluded/rrg>. [Accessed February 2015].
- [15] IRTF, "rrg Discussion Archive - Date Index," [Online]. Available: <http://www.ietf.org/mail-archive/web/rrg/current/maillist.html>. [Accessed February 2015].
- [16] IETF, "Open discussion forum for long/wide-range architectural issues Discussion Archive - Date Index," [Online]. Available: <http://www.ietf.org/mail-archive/web/architecture-discuss/current/maillist.html>. [Accessed February 2015].
- [17] T. Li, "RFC 6227: Design Goals for Scalable Internet Routing," May 2011. [Online]. Available: <http://tools.ietf.org/html/rfc6227>.
- [18] D. Meyer, L. Zhang and K. Fall, "RFC 4984: Report from the IAB Workshop on Routing and Addressing," September 2007. [Online]. Available: <http://tools.ietf.org/html/rfc4984>.
- [19] G. Huston, "BGP Growth Revisited," November 2011. [Online]. Available: <http://www.potaroo.net/ispcol/2011-11/bgp2011.html>.
- [20] G. Huston, "BGP in 2014," January 2015. [Online]. Available: <http://www.potaroo.net/ispcol/2015-01/bgp2014.html>.
- [21] G. Huston, "Addressing 2014 - And then there were 2!," January 2015. [Online]. Available: <http://www.potaroo.net/ispcol/2015-01/addressing2014.html>.
- [22] Y. Rekhter, T. Li and S. Hares, "RFC 4271: A Border Gateway Protocol 4 (BGP-4)," January 2006. [Online]. Available: <http://tools.ietf.org/html/rfc4271>.
- [23] G. Huston, "BGP Reports - BGP Table Data," 7 August 2013. [Online]. Available: <http://bgp.potaroo.net/index-bgp.html>.
- [24] D. Mowery and T. Simcoe, "Is the Internet a US invention?—an economic and technological history of computer networking," *Research Policy*, vol. 31, no. 8-9, pp. 1369-1387, December 2002.
- [25] M. Boucadair and D. Binet, *Solutions for Sustaining Scalability in Internet Growth*, France: IGI Global, 2014.
- [26] B. Carpenter, J. Crowcroft and Y. Rekhter, "RFC 2101: IPv4 Address Behaviour Today," February 1997. [Online]. Available: <http://tools.ietf.org/html/rfc2101>.
- [27] C. J. Bennet, S. W. Edge and A. J. Hinchley, "IEN #1: Issues in the Interconnection of Datagram Networks," 29 July 1977. [Online]. Available: <http://www.postel.org/ien/pdf/ien001.pdf>.
- [28] D. Massey, L. Wang, B. Zhang and L. Zhang, "A scalable routing system design for future internet," in *Proceedings of ACM SIGCOMM Workshop on IPv6*, Kyoto, Japan, 2007.
- [29] S. Brim, "LISP Analysis," March 2008. [Online]. Available: <https://tools.ietf.org/html/draft-brim-lisp-analysis-00>.

- [30] J. Abley, K. Lindqvist, E. Davies, B. Black and V. Gill, "IPv4 Multihoming Practices and Limitations," [Online]. Available: <http://tools.ietf.org/html/rfc4116>.
- [31] T. Bates and Y. Rekhter, "RFC 2260: Scalable Support for Multi-homed Multi-provider Connectivity," January 1998. [Online]. Available: <http://tools.ietf.org/html/rfc2260>.
- [32] Postscapes, "Internet of Things Market Forecast," [Online]. Available: <http://postscapes.com/internet-of-things-market-size>. [Accessed February 2015].
- [33] C. Perkins, "RFC 5944: IP Mobility Support for IPv4, Revised," November 2010. [Online]. Available: <http://tools.ietf.org/html/rfc5944>.
- [34] C. Perkins, D. Johnson and J. Arkko, "RFC 6275: Mobility Support in IPv6," July 2011. [Online]. Available: <https://tools.ietf.org/html/rfc6275>.
- [35] H. Soliman, C. Castelluccia, K. ElMalki and L. Bellier, "RFC 5380: Hierarchical Mobile IPv6 (HMIPv6) Mobility Management," October 2008. [Online]. Available: <https://tools.ietf.org/html/rfc5380>.
- [36] IETF, "Multipath TCP (mptcp)," [Online]. Available: <https://datatracker.ietf.org/wg/mptcp/documents/>. [Accessed February 2015].
- [37] J. Saltzer, D. Reed and D. Clark, "End-to-end arguments in system design," *ACM Transactions on Computer Systems (TOCS)*, vol. 2, no. 4, pp. 277-288, 1984.
- [38] Cisco Systems, Inc., "Document ID 13753: BGP Best Path Selection Algorithm," 21 May 2012. [Online]. Available: http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a0080094431.shtml.
- [39] B. Carpenter, R. Atkinson and H. Flinck, "Renumbering Still Needs Work," May 2010. [Online]. Available: <http://tools.ietf.org/html/rfc5887>.
- [40] G. Huston, "IPv6: IPv6 / IPv4 Comparative Statistics," [Online]. Available: <http://bgp.potaroo.net/v6/v6rpt.html>. [Accessed 22 August 2013].
- [41] G. Huston, "The BGP World is flat," November 2011. [Online]. Available: <http://www.potaroo.net/ispcol/2011-12/flat.html>. [Accessed July 2015].
- [42] J. Durand, I. Pepelnjak and G. Doering, "RFC 7454: BGP Operations and Security," February 2015. [Online]. Available: <https://tools.ietf.org/html/rfc7454>.
- [43] J. Shoch, "IEN #19: A note on Inter-Network Naming, Addressing, and Routing," XEROX PARC, January 1978. [Online]. Available: <http://www.postel.org/ien/pdf/ien019.pdf>.
- [44] C. Sunshine, "IEN #178: Addressing Problems in Multi-Network Systems," University of Southern California, April 1981. [Online]. Available: <http://www.postel.org/ien/pdf/ien178.pdf>.
- [45] J. Saltzer, "Name Binding of Objects," Massachusetts Institute of Technology, 1978. [Online]. Available: web.mit.edu/Saltzer/www/publications/nbo/nbo.pdf.

- [46] J. N. Chiappa, "Endpoints and Endpoint Names: A Proposed Enhancement to the Internet Architecture," 1999. [Online]. Available: <http://www.chiappa.net/~jnc/tech/endpoints.txt>.
- [47] J. Munkres, *Topology: A First Course*, Prentice Hall College Div, 1974.
- [48] ISO, "Information technology – Open Systems Interconnection – Basic Reference Model: Naming and addressing". Patent ISO/IEC 7498-3:1997, 1997.
- [49] J. Saltzer, "On the Naming and Binding of Network Destinations," *Local Computer Networks*, pp. 311-317, August 1982.
- [50] R. Hinden and S. Deering, "RFC 4291: IP Version 6 Addressing Architecture," February 2006. [Online]. Available: <http://tools.ietf.org/html/rfc4291>.
- [51] M. O'Dell, "GSE: The Alternative Addressing Architecture for IPv6," February 1997. [Online]. Available: <http://tools.ietf.org/html/draft-ietf-ipngwg-gseaddr-00>.
- [52] F. Baker, E. Lear and R. Droms, "RFC 4192: Procedures for Renumbering an IPv6 Network without a Flag Day," [Online]. Available: <http://tools.ietf.org/html/rfc4192>.
- [53] T. Li, "RFC 6115: Recommendation for a Routing Architecture," February 2011. [Online]. Available: <http://tools.ietf.org/html/rfc6115>.
- [54] R. Hinden, "RFC 1955: New Scheme for Internet Routing and Addressing (ENCAPS) for IPng," June 1996. [Online]. Available: <http://tools.ietf.org/html/rfc1955>.
- [55] R. Smart and D. Clark, "[RRG] GSE History," January 1995. [Online]. Available: <http://www.ietf.org/mail-archive/web/rrg/current/msg02455.html>.
- [56] D. Jen, M. Meisel, H. Yan, D. Massey, L. Wang, B. Zhang and L. Zhang, "Towards A New Internet Routing Architecture: Arguments for Separating Edges from Transit Core," 2008.
- [57] R. Moskowitz and P. Nikander, "RFC 4423: Host Identity Protocol (HIP) Architecture," May 2006. [Online]. Available: <http://tools.ietf.org/html/rfc4423>.
- [58] R. Moskowitz, P. Nikander, P. Jokela and T. Henderson, "RFC 5201: Host Identity Protocol," April 2008. [Online]. Available: <http://tools.ietf.org/html/rfc5201>.
- [59] E. Nordmark and M. Bagnulo, "RFC 5533: Shim6: Level 3 Multihoming Shim Protocol for IPv6," June 2009. [Online]. Available: <http://tools.ietf.org/html/rfc5533>.
- [60] X. Xu, "Hierarchical Routing Architecture," in *4th Euro-NGI Conference*, Krakow, Poland, April, 2008.
- [61] X. Xu, "Routing Architecture for the Next Generation Internet (RANGI)," August 2010. [Online]. Available: <http://tools.ietf.org/html/draft-xu-rangi-04>.
- [62] X. Xu, "Transition Mechanisms for Routing Architecture for the Next Generation Internet (RANGI)," July 2009. [Online]. Available: <http://tools.ietf.org/html/draft-xu-rangi-proxy-01>.
- [63] R. Whittle, "Ivip Page," [Online]. Available: <http://www.firstpr.com.au/ip/ivip/>.

- [64] R. Whittle, "Glossary of some Ivip and scalable routing terms," March 2010. [Online]. Available: <http://tools.ietf.org/html/draft-whittle-ivip-glossary-01>.
- [65] R. Whittle, "Ivip (Internet Vastly Improved Plumbing) Architecture," March 2010. [Online]. Available: <http://tools.ietf.org/html/draft-whittle-ivip-arch-04>.
- [66] R. Whittle, "Ivip4 ETR Address Forwarding," July 2010. [Online]. Available: <http://tools.ietf.org/html/draft-whittle-ivip-etr-addr-forw-01>.
- [67] R. Whittle and S. Russert, "TTR Mobility Extensions for Core-Edge Separation Solutions to the Internet's Routing Scaling Problem," October 2011. [Online]. Available: <http://www.firstpr.com.au/ip/ivip/TTR-Mobility.pdf>.
- [68] R. Whittle, "IPTM - Ivip's approach to solving the problems with encapsulation overhead, MTU, fragmentation and Path MTU Discovery," January 2010. [Online]. Available: <http://www.firstpr.com.au/ip/ivip/pmtud-frag/>.
- [69] R. Whittle, "DRTM - Distributed Real Time Mapping for Ivip and LISP," March 2010. [Online]. Available: <https://tools.ietf.org/html/draft-whittle-ivip-drtm-01>.
- [70] P. Frejborg, "RFC 6306: Hierarchical IPv4 Framework," July 2011. [Online]. Available: <http://tools.ietf.org/html/rfc6306>.
- [71] Y. Wang, W. Zhang and J. Bi, "Name overlay (NOL) Service for Improving Internet Routing Scalability," Venice, Italy, July, 2010.
- [72] M. Menth, M. Hartmann and D. Klein, "Global Locator, Local Locator, and Identifier Split (GLI-Split)," April 2010. [Online]. Available: <http://www3.informatik.uni-wuerzburg.de/TR/tr470.pdf>.
- [73] M. Menth, M. Hartmann and D. Klein, "Global Locator, Local Locator, and Identifier Split (GLI-Split)," *Future Internet 2013*, vol. V, no. 1, pp. 67-94, January, 2013.
- [74] J. Adan, "Tunneled Inter-domain Routing (TIDR)," April 2007. [Online]. Available: <http://tools.ietf.org/html/draft-adan-idr-tidr-01>.
- [75] J. Adan, "TIDR using the IDENTIFIERS attribute," April 2007. [Online]. Available: <http://www.ietf.org/mail-archive/web/ram/current/msg01308.html>.
- [76] J. Adan, "LISP etc architecture," December 2007. [Online]. Available: <http://www.ietf.org/mail-archive/web/rrg/current/msg00869.html>.
- [77] R. Atkinson, S. Bhatti, S. Hailes, D. Rehunathan and M. Lad, "ILNP - Identifier-Locator Network Protocol," June 2011. [Online]. Available: <http://ilnp.cs.st-andrews.ac.uk>.
- [78] R. Atkinson, "ILNP Concept of Operations," July 2011. [Online]. Available: <http://tools.ietf.org/html/draft-rja-ilnp-intro-11>.

- [79] R. Atkinson and S. Bhatti, "RFC 6740: Identifier-Locator Network Protocol (ILNP) Architectural Description," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6740>.
- [80] R. Atkinson and S. Bhatti, "RFC 6741: Identifier-Locator Network Protocol (ILNP) Engineering Considerations," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6741>.
- [81] R. Atkinson, S. Bhatti and S. Rose, "RFC 6742: DNS Resource Records for ILNP," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6742>.
- [82] R. Atkinson and S. Bhatti, "RFC 6743: ICMP Locator Update Message for the Identifier-Locator Network Protocol for IPv6 (ILNPv6)," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6743>.
- [83] R. Atkinson and S. Bhatti, "RFC 6744: ILNP Nonce Destination Option," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6744>.
- [84] R. Atkinson and S. Bhatti, "RFC 6745: ICMP Locator Update Message," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6745>.
- [85] R. Atkinson, "RFC 6746: IPv4 Options for the Identifier-Locator Network Protocol (ILNP)," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6746>.
- [86] R. Atkinson and S. Bhatti, "RFC 6747: Address Resolution Protocol (ARP) for the Identifier-Locator Network Protocol for IPv4 (ILNPv4)," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6747>.
- [87] R. Atkinson and S. Bhatti, "RFC 6748: Optional Advanced Deployment Scenarios for the Identifier-Locator Network Protocol (ILNP)," November 2012. [Online]. Available: <http://tools.ietf.org/html/rfc6748>.
- [88] B. Zhang and L. Zhang, "Evolution Towards Global Routing Scalability," October 2009. [Online]. Available: <http://tools.ietf.org/html/draft-zhang-evolution-02>.
- [89] B. Zhang, L. Wang, X. Zhao, Y. Liu and L. Zhang, "An Evaluation Study of Router FIB Aggregatability," [Online]. Available: <http://www.ietf.org/proceedings/76/slides/grow-2.pdf>.
- [90] L. Zhang, P. Francis, X. Xu, H. Ballani, D. Jen and R. Raszuk, "Virtual Aggregation (VA)," November 2009. [Online]. Available: <http://www.ietf.org/proceedings/76/slides/grow-5.pdf>.
- [91] C. Vogt, "Simplifying Internet Applications Development With A Name-Based Sockets Interface," December 2009. [Online]. Available: <http://christianvogt.mailup.net/pub/vogt-2009-name-based-sockets.pdf>.
- [92] J. Ubillos, M. Xu, Z. Ming and C. Vogt, "Name-Based Sockets Architecture," September 2010. [Online]. Available: <http://tools.ietf.org/html/draft-ubillos-name-based-sockets-03>.

- [93] D. Jen, M. Meisel, H. Yan, D. Massey, L. Wang, B. Zhang and L. Zhang, "APT: A Practical Transit Mapping Service," November 2007. [Online]. Available: <http://tools.ietf.org/html/draft-jen-apt-01>.
- [94] F. Templin, "RFC 5320: The Subnetwork Encapsulation and Adaptation Layer (SEAL)," January 2011. [Online]. Available: <http://tools.ietf.org/html/rfc5320>.
- [95] F. Templin, "RFC 5720: Routing and Addressing in Networks with Global Enterprise Recursion (RANGER)," February 2010. [Online]. Available: <http://tools.ietf.org/html/rfc5720>.
- [96] W. Herrin, "Tunneling Route Reduction Protocol (TRRP)," [Online]. Available: <http://bill.herrin.us/network/trrp.html>.
- [97] C. Vogt, "Six/One Router: A Scalable and Backwards Compatible Solution for Provider-Independent Addressing," Seattle, USA, 2008.
- [98] D. Farinacci, V. Fuller, D. Meyer and D. Lewis, "RFC 6830: The Locator/ID Separation Protocol (LISP)," January 2013. [Online]. Available: <http://tools.ietf.org/html/rfc6830>.
- [99] D. Lewis, D. Meyer, D. Farinacci and V. Fuller, "RFC 6832: Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites," January 2013. [Online]. Available: <http://tools.ietf.org/html/rfc6832>.
- [100] V. Fuller, "RFC 6833: Locator/ID Separation Protocol (LISP) Map-Server Interface," January 2013. [Online]. Available: <http://tools.ietf.org/html/rfc6833>.
- [101] L. Iannone, D. Saucez and O. Bonaventure, "RFC 6834: Locator/ID Separation Protocol (LISP) Map-Versioning," January 2013. [Online]. Available: <http://tools.ietf.org/html/rfc6834>.
- [102] J. Curran, S. Brim, D. Farinacci and D. Meyer, "EID Mappings Multicast Across Cooperating Systems for LISP," November 2007. [Online]. Available: <http://tools.ietf.org/html/draft-curran-lisp-emacs-00>.
- [103] E. Lear, "RFC 6837: NERD - A Not-so-novel Endpoint ID (EID) to Routing Locator (RLOC) Database," January 2013. [Online]. Available: <http://tools.ietf.org/html/rfc6837>.
- [104] D. Meyer, S. Brim, N. Chiappa, D. Farinacci, V. Fuller and D. Lewis, "LISP-CONS: A Content distribution Overlay Network Service for LISP," April 2008. [Online]. Available: <http://tools.ietf.org/html/draft-meyer-lisp-cons-04>.
- [105] V. Fuller, D. Farinacci, D. Meyer and D. Lewis, "RFC 6836: Locator/ID Separation Protocol Alternative Logical Topology (LISP+ALT)," January 2013. [Online]. Available: <http://tools.ietf.org/html/rfc6836>.
- [106] V. Fuller, D. Lewis, D. Ermagan and A. Jain, "draft-fuller-lisp-ddt: LISP Delegated Database Tree," September 2012. [Online]. Available: <http://tools.ietf.org/html/draft-fuller-lisp-ddt>.

- [107] I. Stoica, R. Morris, D. Karger, F. Kaashoek and H. Balakrishnan, May 2013. [Online]. Available: <https://github.com/sit/dht/wiki>.
- [108] L. Mathy and L. Iannone, "LISP-DHT: Towards a DHT to map identifiers onto locators," in *Proceedings of the 2008 ACM CoNEXT Conference*, Madrid, Spain, 2008.
- [109] R. Froom, E. Frahim and B. Sivasubramanian, CCNP Self-Study: Understanding and Configuring Multilayer Switching, 4th Edition ed., Cisco Press, 2007.
- [110] IETF, January 2014. [Online]. Available: <https://datatracker.ietf.org/wg/lisp/charter/>.
- [111] L. Iannone, October 2011. [Online]. Available: <http://www.openlisp.org/>.
- [112] Cisco Systems, Inc., November 2013. [Online]. Available: <http://lisp.cisco.com/>.
- [113] UPC BarcelonaTECH, 2014. [Online]. Available: <http://lispmob.org/home>.
- [114] D. Farinacci, D. Lewis, D. Meyer and C. White, "draft-meyer-lisp-mn: LISP Mobile Node," July 2013. [Online]. Available: <http://tools.ietf.org/html/draft-meyer-lisp-mn>.
- [115] AVM GmbH, "Unterstützung für das Locator Identifier Separation Protocol (LISP)," 2013. [Online]. Available: http://www.avm.de/de/Service/Service-Portale/Service-Portal/Sonstige_Dokumente/labor_lisp.php.
- [116] LISP4/LISP6.net, "LISP BetaNetwork," 2011. [Online]. Available: <http://www.lisp4.net/beta-network/>.
- [117] American Registry for Internet Numbers, "ASN3943," January 2014. [Online]. Available: <http://whois.arin.net/rest/asn/AS3943/pft>.
- [118] D. Farinacci and D. Meyer, "RFC 6835: The Locator/ID Separation Protocol Internet Groper (LIG)," January 2013. [Online]. Available: <http://tools.ietf.org/html/rfc6835>.
- [119] UPC BarcelonaTECH, 2014. [Online]. Available: <http://lispmon.net/>.
- [120] F. Coras, A. Cabellos and L. Jakab, "CoreSim: A Simulator for Evaluating LISP Mapping Systems," Cluj-Napoca, 2009.
- [121] A. Cabellos, J. Domingo Pascual, D. Saucez and O. Bonaventure, "Validation of a LISP simulator," 2011. [Online]. Available: <http://upcommons.upc.edu/e-prints/bitstream/2117/14351/1/Cabellos.pdf>.
- [122] D. Klein, M. Hoefling, M. Hartmann and M. Menth, "Integration of LISP and LISP-MN into INET," in *Proceedings of the IEEE 5th International ICST Conference on Simulation Tools and Techniques*, Desenzano del Garda, 2012.
- [123] D. Klein, M. Hartmann and M. Menth, "NAT Traversal for LISP Mobile Node," July 2010. [Online]. Available: <http://tools.ietf.org/html/draft-klein-lisp-mn-nat-traversal>.

- [124] J. Kim, L. Iannone and A. Feldmann, "A deep dive into the LISP cache and what ISPs should know about it," *NETWORKING 2011*, vol. 6640, no. ISBN: 978-3-642-20756-3, pp. 367-378, 2011.
- [125] D. Meyer and D. Lewis, "Architectural Implications of Locator/ID Separation," January 2009. [Online]. Available: <http://tools.ietf.org/html/draft-meyer-loc-id-implications-01>.
- [126] J. Day, "Why Loc/Id Split Isn't the Answer," Pouzin Society, 2008. [Online]. Available: <http://pouzinsociety.org/images/LocIDSplit090309.pdf>.
- [127] R. Hinden, "RFC 3768: Virtual Router Redundancy Protocol (VRRP)," April 2004. [Online]. Available: <https://tools.ietf.org/html/rfc3768>.
- [128] S. Nadas, "RFC 5798: Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6," March 2010. [Online]. Available: <https://tools.ietf.org/html/rfc5798>.
- [129] D. Saucez, O. Bonaventure, L. Iannone and C. Filsfils, "LISP ITR Graceful Restart," December 2013. [Online]. Available: <https://tools.ietf.org/html/draft-saucez-lisp-itr-graceful-03>.
- [130] D. Saucez, J. Kim, L. Iannone, O. Bonaventure and C. Filsfils, "A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers," *NETWORKING 2012*, vol. 7289, pp. 397-408, 2012.
- [131] V. Veselý, M. Marek, O. Ryšavý and M. Švéda, "Multicast, TRILL and LISP Extensions for INET," *Journal On Advances in Networks and Services*, vol. 7, no. 3&4, pp. 240-251, 2014.
- [132] V. Veselý and O. Ryšavý, "Locator/Id Split Protocol Improvement for High-Availability Environment," in *Proceedings of The Tenth International Conference on Networking and Services*, Roma, Italy, 2015.
- [133] V. Veselý and O. Ryšavý, "Map-Cache Synchronization and Merged RLOC Probing Study for LISP," *International Journal On Advances in Intelligent Systems*, vol. 8, no. 3&4, 2015.
- [134] Cisco Systems, Inc., "Cisco IOS IP Routing: LISP Command Reference - LISP Show Commands [Support]," [Online]. Available: http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_lisp/command/ip-lisp-cr-book/ip-lisp-cr-book_chapter_01011.html. [Accessed February 2014].
- [135] J. Day, "RINARefModelPart1-0 130925: Part 1 - Basic Concepts of Distributed Systems," Pouzin Society, 2013.
- [136] J. Day, "RINARefModelPart2-1 130925: Part 2 - Distributed Applications, Chapter 1 - Basic Concepts of Distributed Applications," Pouzin Society, 2013.
- [137] J. Day and E. Trouva, "RINARefModelPart2-2 140102: Part 2 - Distributed Applications, Chapter 2 - Introduction to Distributed Management Systems," Pouzin Society, 2014.
- [138] J. Day, "RINARefModelPart3-1 140102: Part 3 - Distributed InterProcess Communication, Chapter 1 - Fundamental Structure," Pouzin Society, 2012.

- [139] J. Day, "RINARefModelPart3-2 140102: Part 3 - Distributed InterProcess Communication, Chapter 2 - DIF Operations," Pouzin Society, 2012.
- [140] J. Day, "An introduction to the Recursive InterNetwork Architecture," January 2015. [Online]. Available: <http://ict-pristine.eu/wp-content/uploads/2014/12/GhentIntroRINAPt1-150119.pdf>. [Accessed April 2015].
- [141] J. Day, I. Matta and K. Mattar, "Networking is IPC: a guiding principle to a better internet," in *CoNEXT '08 Proceedings of the 2008 ACM CoNEXT Conference*, New York, NY, USA, 2008.
- [142] R. Watson, "The Delta-t transport protocol: features and experience," in *Proceedings 14th Conference on Local Computer Networks*, Minneapolis, USA, 1989.
- [143] E. Trouva, E. Grasa, J. Day and S. Bunch, "Layer discovery in RINA networks," in *IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Barcelona, Spain, 2012.
- [144] J. Day, "D-Base-2011-017: IPC Resource Manager (IRM) Specification," Pouzin Society, 2012.
- [145] J. Day, "D-Base-2012-014: Basic Enrollment Specification," Pouzin Society, 2012.
- [146] J. Day, "D-Base-2010-007: Delimiting Module," Pouzin Society, 2009.
- [147] J. Day, "DelimitingGeneral130904: Delimiting Module," Pouzin Society, 2013.
- [148] J. Day, M. Marek, L. Bergesio and M. Tarzan, "EFCPSpec140824_MT_LBJD_MM_v6.6: Error and Flow Control Protocol Specification, Data Transfer + Data Transfer Control," Pouzin Society, 2015.
- [149] J. Day, "D-Base-2012-010: Relaying and Multiplexing Task Specification," Pouzin Society, 2012.
- [150] J. Day, "D-Base-2011-015: Flow Allocator Specification," Pouzin Society, 2011.
- [151] J. Day, "RINA-RFC-2010-002: Notes on the Resource Allocator," Pouzin Society, 2010.
- [152] E. Grasa, S. Bunch and P. deWolf, "Specification of Managed Objects for the Demo DIF," Pouzin Society, 2012.
- [153] J. Day, "Notes on the OIB/RIB Daemon," Pouzin Society, 2010.
- [154] ISO, "Information technology – Open Systems Interconnection – Service definition for the Application Service Object Association Control Service Element". Patent ISO/IEC 15953:1999, 1999.
- [155] ISO, "Information technology – Open Systems Interconnection – Connectionless protocol for the Association Control Service Element: Protocol specification". Patent ISO/IEC 10035-1:1995, 1995.
- [156] ISO, "Information technology – Open Systems Interconnection – Common Management Information Protocol: Specification". Patent ISO/IEC 9596-1:1998, 1997.

- [157] S. Bunch, "D-Base-2010-009: CDAP – Common Distributed Application Protocol," Pouzin Society, 2010.
- [158] S. Bunch, J. Day and E. Trouva, "D-Base-2012-016: Common Application Connection Establishment Phase (CACEP)," Pouzin Society, 2012.
- [159] IRATI consortium, "IRATI Investigating RINA as an Alternative to TCP/IP," 2015. [Online]. Available: <http://irati.eu/>. [Accessed July 2015].
- [160] IRINA consortium, "IRINA," 2015. [Online]. Available: <http://www.geant.net/opencall/Optical/Pages/IRINA.aspx>. [Accessed July 2015].
- [161] PRISTINE consortium, "PRISTINE | PRISTINE will take a major step forward in the integration of networking and distributed computing," 2015. [Online]. Available: <http://ict-pristine.eu/>. [Accessed July 2015].
- [162] IRATI consortium, "IRATI · GitHub," [Online]. Available: <https://github.com/IRATI>. [Accessed September 2015].
- [163] J. Day, E. Grasa, S. Bunch and P. deWolf, "RINA-2012-005: Specification for shim IPC Processes over IP Layers," Pouzin Society, 2012.
- [164] IRATI consortium, "Specification for shim IPC Processes over 802.1Q," Pouzin Society, 2013.
- [165] Boston University, "Boston University's prototype of the RINA architecture · GitHub," [Online]. Available: <https://github.com/ProtoRINA>. [Accessed September 2015].
- [166] Y. Wang, I. Matta, F. Esposito and J. Day, "Introducing ProtoRINA: A Prototype for Programming Recursive-Networking Policies," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 129-131, 2014.
- [167] V. Veselý, M. Marek, T. Hykel and K. Rausch, "Deliverable 2.4: RINA Simulator, basic functionality," January 2015. [Online]. Available: <http://ict-pristine.eu/wp-content/uploads/2013/12/PRISTINE-D24-RINASim-draft.pdf>. [Accessed July 2015].
- [168] Brno University of Technology, "kvetak/RINA," GitHub, 2014. [Online]. Available: <https://github.com/kvetak/RINA>. [Accessed July 2015].
- [169] PRISTINE consortium, "RINASimulator / RINA Sim Code," Open Source Projects, 2014. [Online]. Available: <https://opensourceprojects.eu/p/pristine/rinasimulator/>. [Accessed July 2015].
- [170] OpenSim Ltd., "OMNeT++ - Manual version 4.6," 2015. [Online]. Available: <https://omnetpp.org/doc/omnetpp/manual/usman.html#sec534>. [Accessed July 2015].
- [171] V. Veselý, M. Marek, T. Hykel and O. Ryšavý, "Skip This Paper - RINASim: Your Recursive InterNetwork Architecture Simulator," in *Proceedings of the 2nd OMNeT++ Community Summit*, Zurich, Switzerland, 2015.

- [172] V. Veselý, "Deliverable 2.6: RINA Simulator, advanced functionality," November 2015. [Online]. [Accessed November 2015].
- [173] L. Jakab, A. Cabellos-Aparicio, F. Coras, J. Domingo-Pascual and D. Lewis, "RFC 7215: Locator/Identifier Separation Protocol (LISP) Network Element Deployment Considerations," April 2014. [Online]. Available: <https://tools.ietf.org/html/rfc7215>.
- [174] D. Saucez, L. Iannone and O. Bonaventure, "LISP Threats Analysis," August 2015. [Online]. Available: <https://tools.ietf.org/html/draft-ietf-lisp-threats-13>.
- [175] Pouzin Society, "The Pouzin Society - Building A Better Network," 2012. [Online]. Available: <http://www.pouzinsociety.org/>. [Accessed November 2015].
- [176] PRISTINE consortium, "Deliverable 3.2: Initial specification and proof of concept implementation of techniques to enhance performance and resource utilization in networks," April 2015. [Online]. Available: http://ict-pristine.eu/wp-content/uploads/2013/12/pristine-d32-enhance-performance-and-resource-utilization-in-networks-v1_0.pdf. [Accessed September 2015].
- [177] F. Hrizi, A. Laouiti and H. Chaouchi, "SFR: Scalable Forwarding with RINA for Distributed Clouds," in *Proceedings of 6th International Conference On Network of the Future (NoF 2015)*, Montreal, Canada, 2015.
- [178] V. Veselý, O. Ryšavý and M. Švéda, "IPv6 Unicast and IPv4 Multicast Routing in OMNeT++," in *Proceedings of the IEEE 6th International ICST Conference on Simulation Tools and Techniques*, Cannes, France, 2013.
- [179] V. Veselý, O. Ryšavý and M. Švéda, "Protocol Independent Multicast in OMNeT++," in *The Tenth International Conference on Networking and Services*, Chamonix, France, 2014.
- [180] R. Callon, "RFC 1925: The Twelve Networking Truths," 1st April 1996. [Online]. Available: <https://tools.ietf.org/html/rfc1925>.

8 Addendum

8.1 Formats of LISP Control Messages

8.1.1 LISP Map-Request

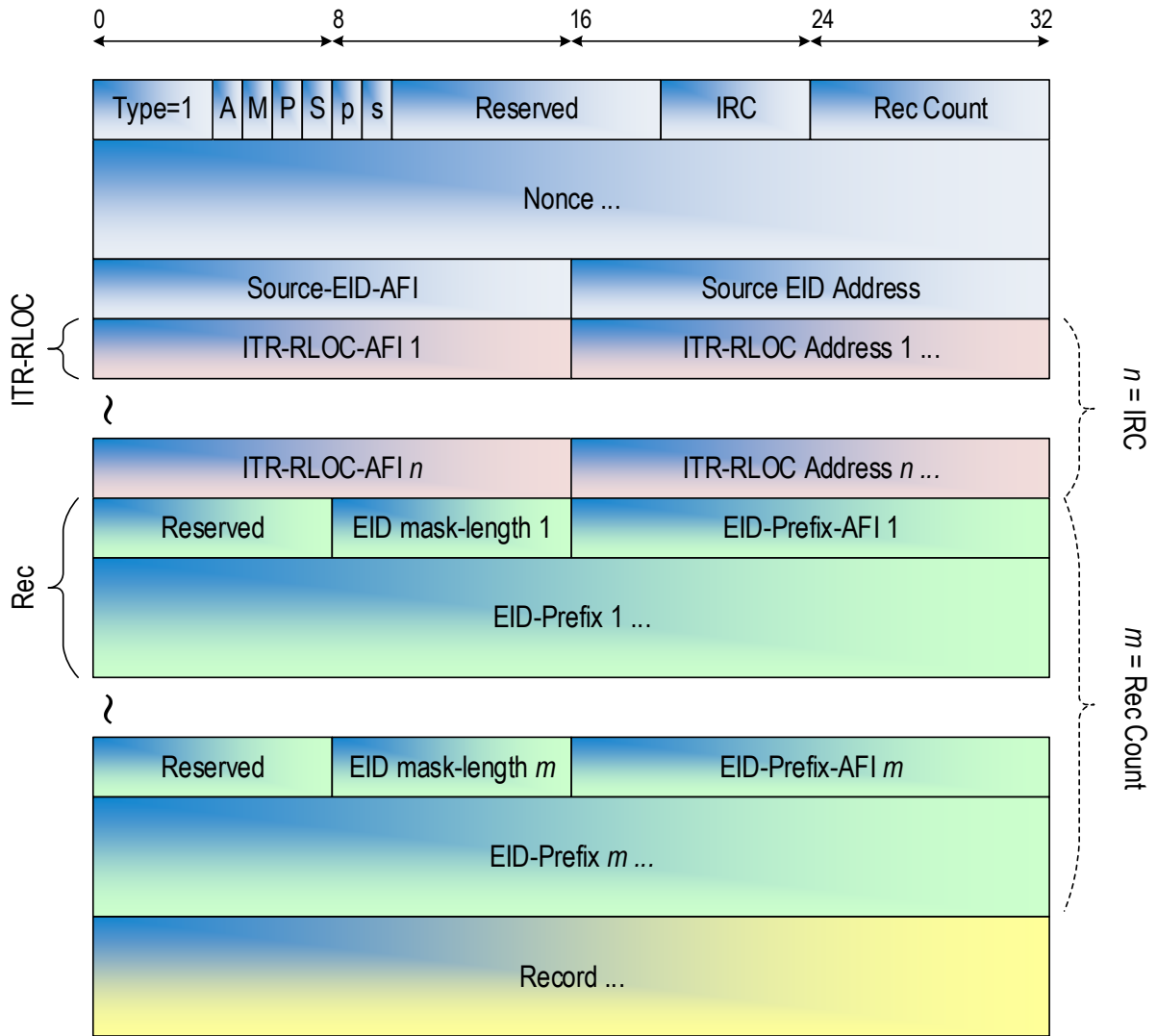


Fig. 74: LISP Map-Request message format

8.1.2 LISP Map-Response

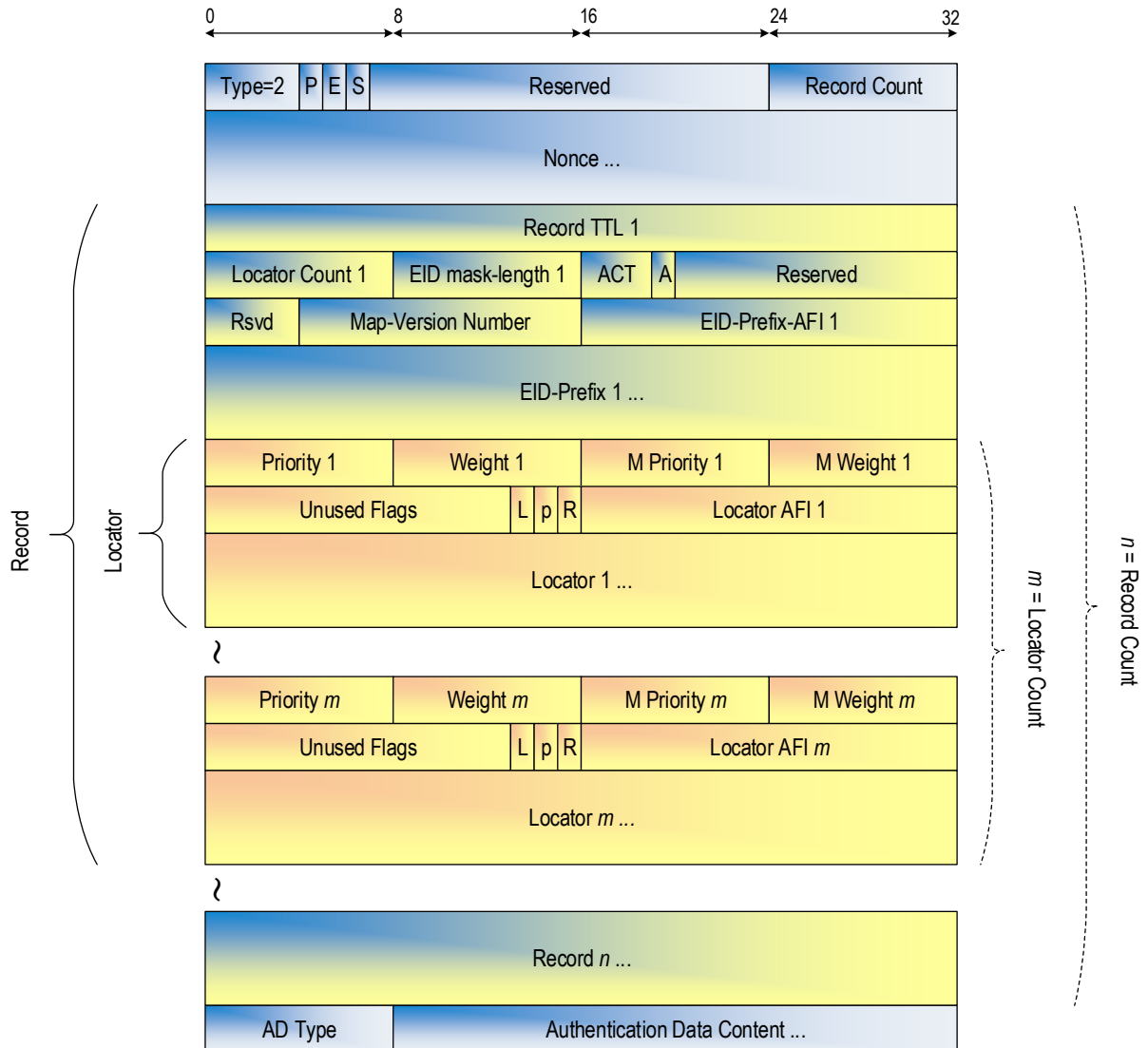


Fig. 75: LISP Map-Reply message format

8.1.3 LISP Map-Register and LISP Map-Notify

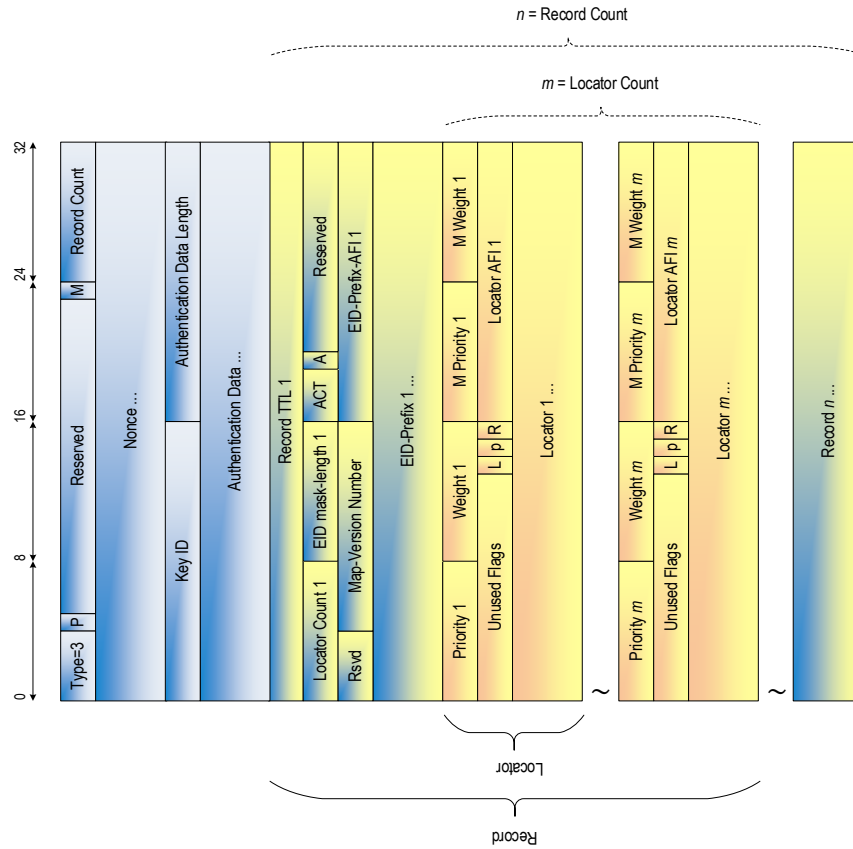


Fig. 76: LISP Map-Register message format

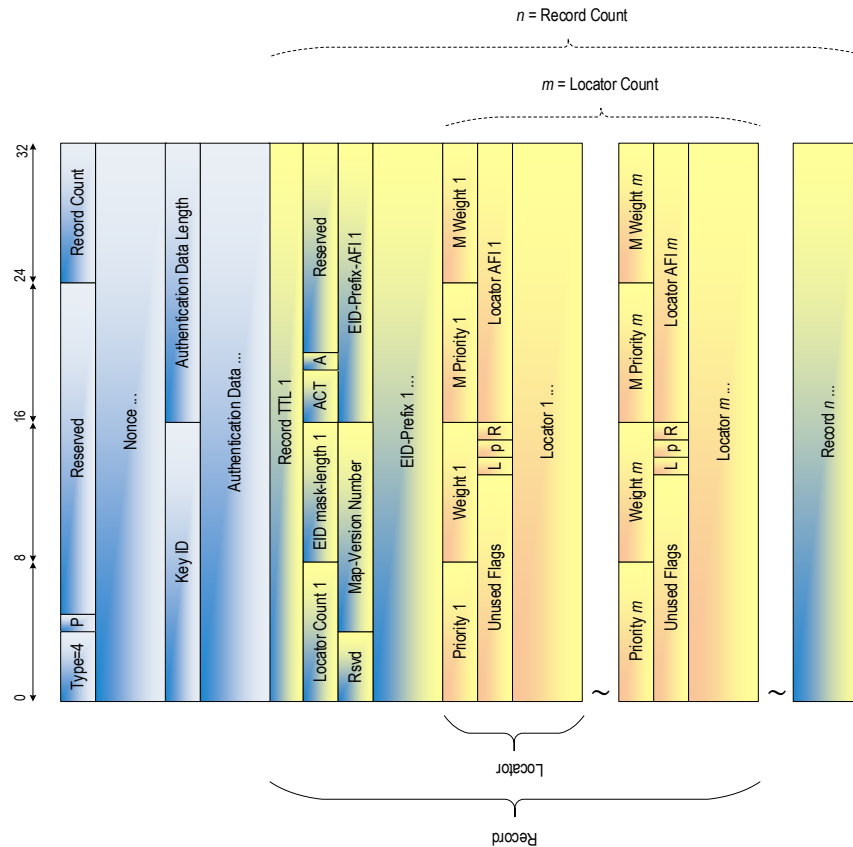


Fig. 77: LISP Map-Notify message format

8.2 ANSARouter Module

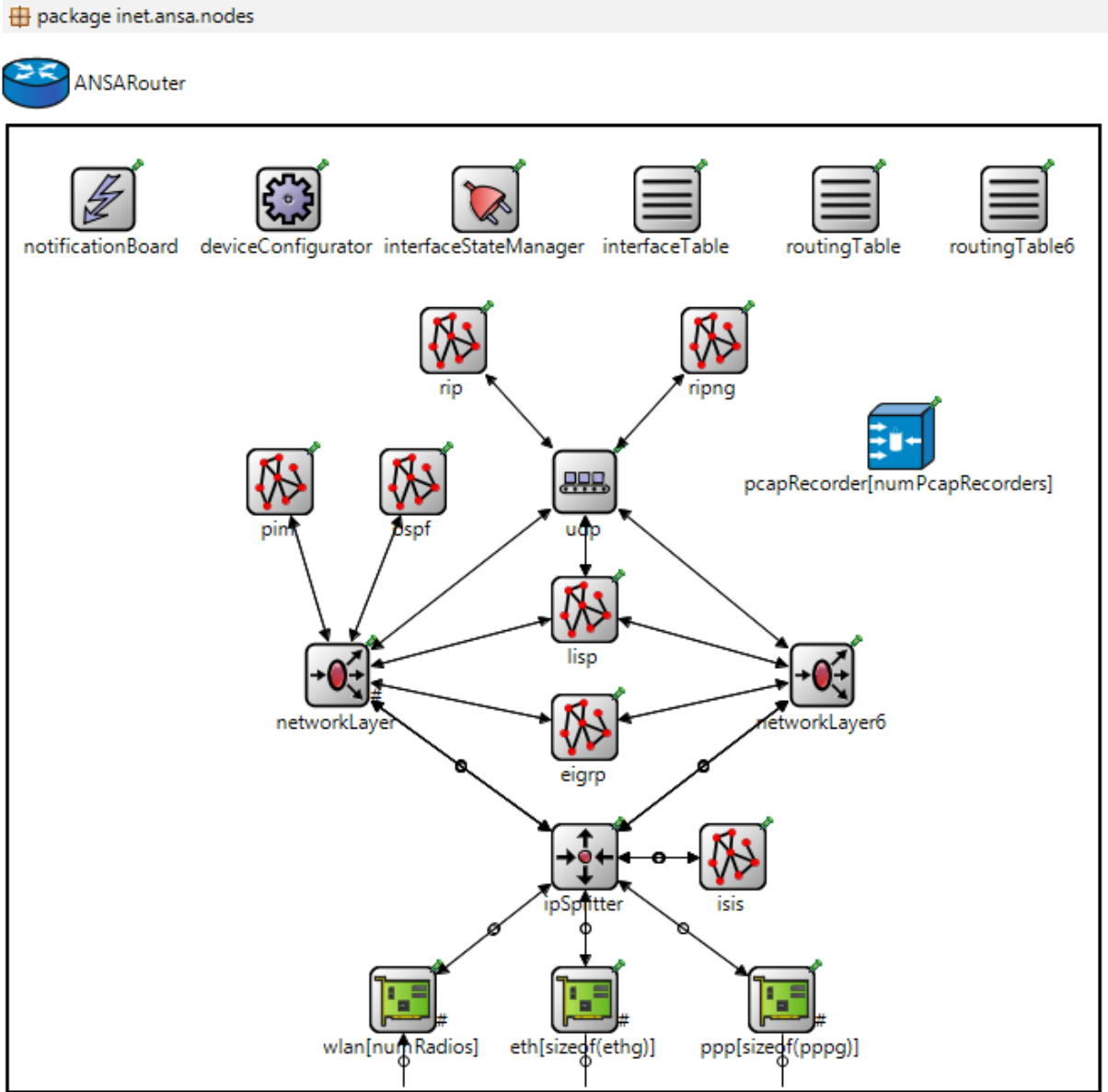


Fig. 78: ANSARouter module structure

8.3 Additional Graphs

8.3.1 Map-Cache Sync Scenario with Single xTR1 Outage

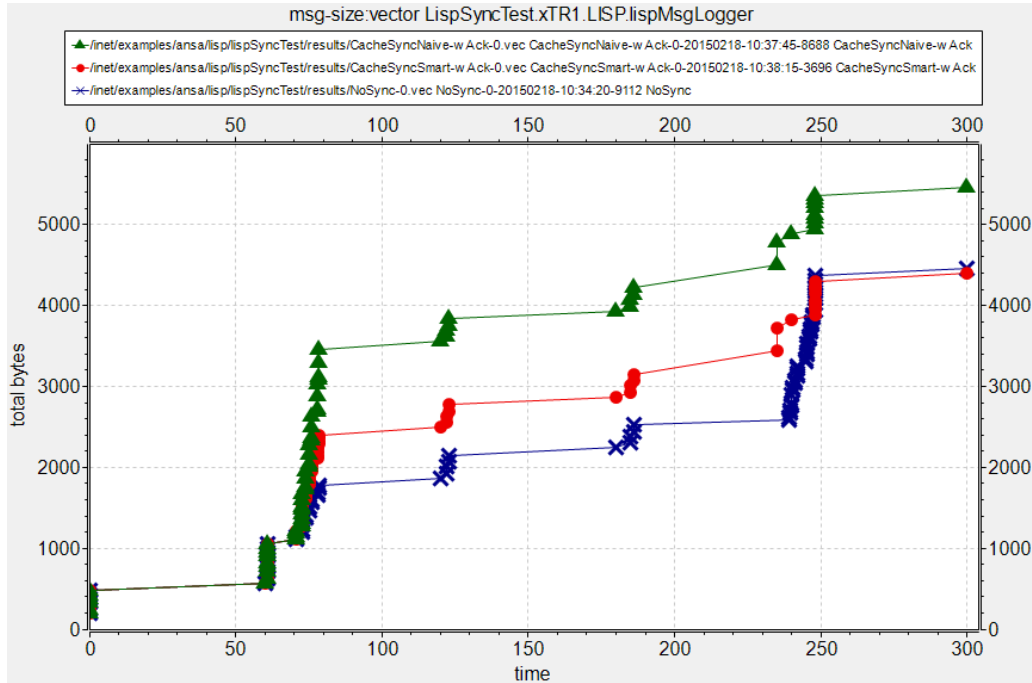


Fig. 79: xTR1's LISP control messages occurrence and total processed byte size in scenario with single outages + ack

8.3.2 Map-Cache Sync Scenario with Three xTR1 Outages

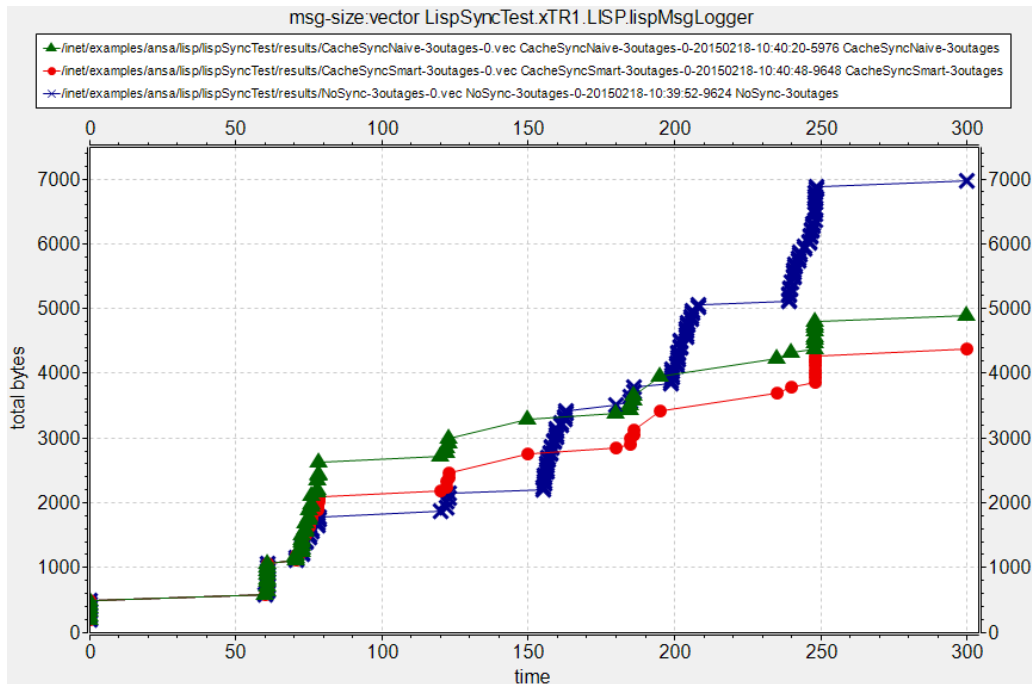


Fig. 80: xTR1's LISP control messages occurrence and total processed byte size in scenario with three outages

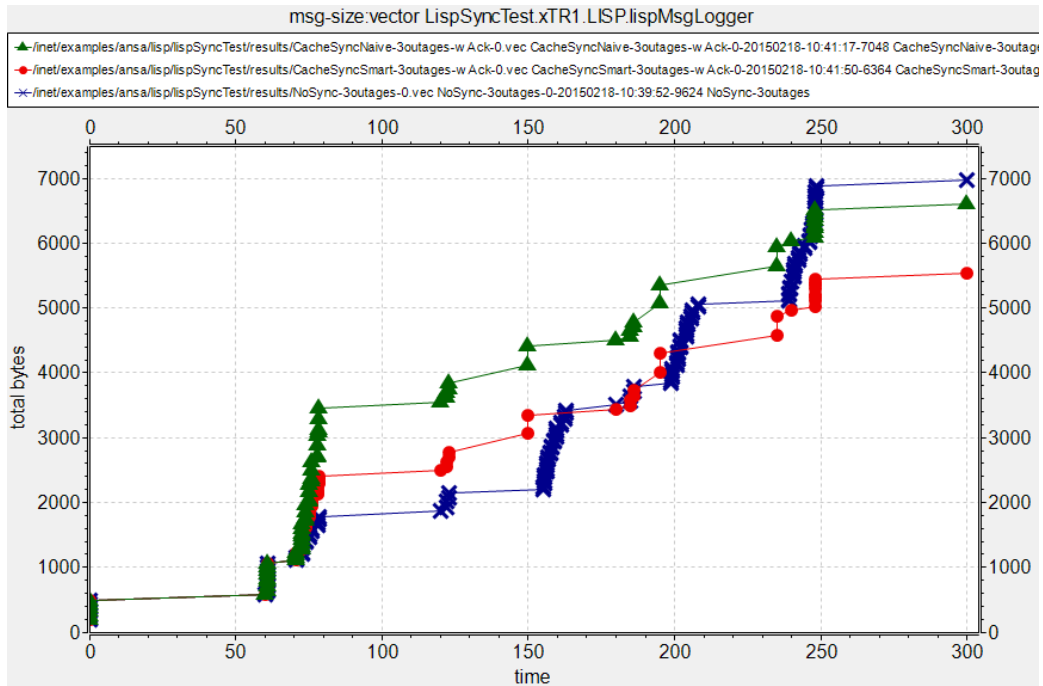


Fig. 81: xTR1's LISP control messages occurrence and total processed byte size in scenario with two outages + ack

8.3.3 RLOC Probe Scenario with Eighty EIDs

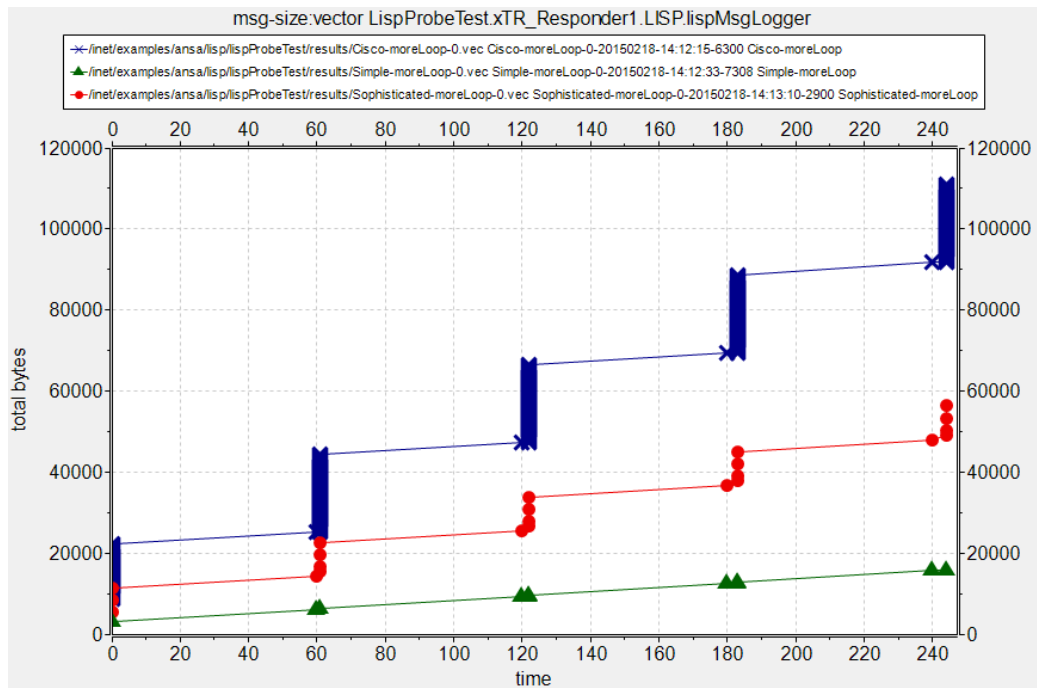


Fig. 82: xTR_Responder1's LISP messages occurrence and total processed byte size in scenario with eighty EIDs

8.4 John Day about RINA

-----Original Message-----

From: John Day
Sent: 26. prosince 2015 22:47
To: Vladimír Veselý
Cc: John Day
Subject: Comments on Section 5

Vesely,

[...]

I have a major comment on the first page of Chapter 5. It isn't in the attached document it is here. I believe that this is very important.

RINA is not a clean-slate architecture, nor was it developed to replace the Internet, nor is it John Day's ideas. In a very real sense, RINA is a continuation of the original internetworking ideas from the mid-1970s. The Internet is the aberration caused by having too much money and making it all free seduced everyone. We have uncovered pieces of the puzzle over the years. The concept of scope of a layer comes from the work of Elie and Zimmermann on CYCLADES. The idea of an overlay comes from them and INWG; Watson's discoveries on synchronization; making protocols invariant wrt syntax; the AP/AE distinction; the re-discovery of the internet layer, etc.

I merely tried to assemble all of the principles and patterns into a single whole, which turned out to imply a much simpler and powerful implementation.

Just think how many things in the Internet require adding something that simply fall out as a consequence of the structure?

I truly believe that if we had not been so embroiled in the politics of standardization and the push to move this stuff to product quickly, others would have seen these patterns much earlier. This is what people should have been doing over the last 40 years.

I did not start out to do a "future internet architecture." As you have heard me say, I was trying to figure out what I didn't understand. This has been all about finding the really fundamental principles. I didn't see the answer and the write it down. It was the product of digging deeply into things to understand what was really going on. I didn't invent that EFCPs cleave into DTP and DTCP. I did the exercise in of separating mechanism and policy for each function in a protocol. I had no idea what it would tell me when I did it. I didn't do it to prove anything. I did it to learn something. Once I had done it, then it was a matter of seeing what it told me. Do you see the difference? I wasn't trying to find a way to fix something or do something new. I was trying to understand.

Now experience did play a role. For example, we kept finding we needed fragmentation in essentially every layer. It was apparent in the early 70s that if one relayed one had to do error and flow control over the top of it.

Why did some layers have a lot of layer management (network layer) and others didn't (transport). It became obvious that the reason was that all layers did the same complement of functions. The IPC Model confirmed that.

The same kind of logic goes with where everything else is and what it does.

(You won't believe how many times I re-visited what order delimiting and SDU protection were and where in the layer.) Looking carefully at what the problem was telling me about the structure.

Why is this important? We are investigating the fundamental principles. We are doing science. We aren't trying to build the future Internet or a replacement for it. It turns out that what we are figuring out to build does solve that problem too, but that wasn't our goal and it shouldn't be our research goal. The Future Internet is trying to figure out what to build.

Their proposals are their opinions on how to do that. We don't have opinion. We are doing what the problem says. Not what John Day says. You don't have to build a network the way RINA says (obviously) but if you take that route it will be inferior in some way probably in a major way.

To lump us in with either the Future Internet efforts or the Internet patches is to detract from what we are doing. We are trying to do for Networking what Maxwell did for Electricity and Magnetism, not because we

(I) have delusions of grandeur, but because IT IS WHAT A SCIENTIST DOES. We are demonstrating the power of theory. Every insight we have had has come from the clarity the theory (RM) provides, not from implementation. We are doing science, not craft. My hope is that our example will get others to return to doing science.

Do you understand what I am trying to say?

Hope Santa was good to you!!

Take care,

John

8.5 RINASim Policies

Parent/Child Name	Owner	Description
AllocateRetry	FA	What happen when M_CREATE is resent by Flow Allocator?
LimitedRetries		Allocation is discontinued, when retransmit threshold is met.
NewFlowRequest	FA	When new flow is being allocated, how are its requirements mapped to RA QoS-cubes?
ScoreComparer		QoS Cube with best score wins.
MinComparer		QoS Cube with minimal feasibility wins.
AddressComparator	RA	Policy used for determining whether a PDU address matches the IPCP's address.
ExactMatch		Exact address matching.
PrefixMatch		Matching based on the best address prefix match.
PDUFG	RA	PDU Forwarding Generator providing data used by the PDU Forwarding policy.
BiDomainGenerator		Populates forwarding policy with entries on the form samePrefix.Id -> port and distinctPrefix.
LatGenerator		Informs of flow metrics to routing as latency based on (N-1)-QoS-cube instead of hops.
MSimpleGenerator		Informs of flow metrics to routing as hops, populates forwarding policy with all existing best next-hops.
QoSDomainGenerator		Populates forwarding policy with best next-hop per destination + QoS.
SimpleGenerator		Informs of flow metrics to routing as hops.
SingleDomainGenerator		Informs of flow metrics to routing as hops intended for domain based routing.
StaticGenerator		Load forwarding information from XML configuration.
QueueAlloc	RA	(N-1)-port queue allocation strategy.
QueuePerNCU		One queue per (N)-Cherish/Urgency class.
QueuePerNFlow		One queue per (N)-flow.
QueuePerNQoS		One queue per (N)-QoS cube.
SingleQueue		One queue for all traffic.
QueueIDGen	RA	Companion policy to QueueAlloc; returns queue ID for a given PDU or Flow object.
IDPerNCU		Used with QueueAlloc::QueuePerNCU.
IDPerNFlow		Used with QueueAlloc::QueuePerNFlow.
IDPerNQoS		Used with QueueAlloc::QueuePerNQoS.
SingleID		Used with QueueAlloc::SingleQueue.
MaxQueue	RMT	Policy invoked when a queue size grows over its threshold.
DumbMaxQ		Used with Monitor::SmartMonitor. Request drop probability to monitor, drop random on that.
ECNMarker		IF queue size >= threshold THEN apply ECN marking on new PDUs. IF size >= max THEN drop.
ReadRateReducer		IF queue size >= allowed_maximum THEN stop receiving data from input ports.
REDDropper		Used with Monitor::REDMonitor; Random Early Detection implementation.
TailDrop		IF queue size >= allowed_maximum THEN drop new PDUs.
UpstreamNotifier		IF queue size >= allowed_maximum THEN send a notification to the PDU sender.

Monitor	RMT	
BEMonitor		Used with Monitor::SmartMonitor. Best-effort using multiple queues.
DLMonitor		Used with Monitor::SmartMonitor. Dela/Loss monitor implementation.
eDLMonitor		Used with Monitor::SmartMonitor. Enhanced-Dela/Loss monitor implementation.
REDMonitor		Used with MaxQueue::REDDropper; Random Early Detection implementation.
DummyMonitor		No operation.
SmartMonitor		Monitor interface for use with dumbMaxQ/dumbSch, which can be queried for drop probability and next queue.
PDUForwarding	RMT	Policy used to decide where to forward a PDU.
DomainTable		A table with {domain:{prefix, QoS} -> { Table:{dstAddr -> port}, default:port } }.
MiniTable		A table with {dstAddr -> port} mappings.
MultiMiniTable		A table with {dstAddr -> vector<port>} mappings.
QoSTable		A table with {(dstAddr, QoS) -> port} mappings.
SimpleTable		A table with {(dstAddr, QoS) -> port} mappings.
Scheduler	RMT	Policy deciding which (N-1)-port queue should be processed next.
DumbSch		Used with Monitor::SmartMonitor. Queries the monitor for the next queue to serve.
LongestQFirst		Pick the queue which contains the most PDUs.
DomainRouting	Routing	
DV		A distance vector-like domain routing protocol.
LS		A link-state-like domain routing protocol.
DummyRouting	Routing	No operation.
SimpleRouting	Routing	
SimpleDV		A simple distance vector-like protocol.
SimpleLS		A simple link-state-like protocol.

Tab. 14: Implemented RINASim policies

8.6 RINASim Demonstration

8.6.1 omnetpp.ini

```
[General]
network = UseCase5
check-signals = true
sim-time-limit = 5min
debug-on-errors = true
#Application setup
**.HostA.applicationProcess1.apName = "SourceA"
**.HostB.applicationProcess1.apName = "DestinationB"
**.iae.aeName = "MyPing"
**.applicationEntity.aeType = "AEMyPing"

#DIF Naming
**.Host*.ipcProcess1.difName = "TopLayer"
**.BorderRouter*.relayIpc.difName = "TopLayer"
**.HostA.ipcProcess0.difName = "MediumLayerA"
**.BorderRouterA.ipcProcess1.difName = "MediumLayerA"
**.HostB.ipcProcess0.difName = "MediumLayerB"
**.BorderRouterB.ipcProcess1.difName = "MediumLayerB"
**.BorderRouterA.ipcProcess2.difName = "MediumLayerAB"
**.InteriorRouter.relayIpc.difName = "MediumLayerAB"
**.BorderRouterB.ipcProcess2.difName = "MediumLayerAB"
**.BorderRouterA.bottomIpc.difName = "BottomLayerA"
**.InteriorRouter.ipcProcess0.difName = "BottomLayerA"
**.BorderRouterB.bottomIpc.difName = "BottomLayerB"
**.InteriorRouter.ipcProcess1.difName = "BottomLayerB"

#Static IPC Addressing
**.HostA.ipcProcess1.ipcAddress = "hA"
**.HostB.ipcProcess1.ipcAddress = "hB"
**.BorderRouterA.relayIpc.ipcAddress = "rA"
**.BorderRouterB.relayIpc.ipcAddress = "rB"
**.HostA.ipcProcess0.ipcAddress = "ha"
**.BorderRouterA.ipcProcess1.ipcAddress = "ra"
**.HostB.ipcProcess0.ipcAddress = "hb"
**.BorderRouterB.ipcProcess1.ipcAddress = "rb"
**.BorderRouterA.ipcProcess2.ipcAddress = "rA"
**.InteriorRouter.relayIpc.ipcAddress = "rC"
**.BorderRouterB.ipcProcess2.ipcAddress = "rB"
**.BorderRouterA.bottomIpc.ipcAddress = "ra"
**.InteriorRouter.ipcProcess0.ipcAddress = "rc"
**.BorderRouterB.bottomIpc.ipcAddress = "rb"
**.InteriorRouter.ipcProcess1.ipcAddress = "rc"

#DIF Allocator settings
**.HostA.difAllocator.configData = xmldoc("config.xml",
"Configuration/Host[@id='HostA']/DA")
**.HostB.difAllocator.configData = xmldoc("config.xml",
"Configuration/Host[@id='HostB']/DA")
**.BorderRouterA.difAllocator.configData =
xmldoc("config.xml",
"Configuration/Router[@id='BorderRouterA']/DA")
**.BorderRouterB.difAllocator.configData =
xmldoc("config.xml",
"Configuration/Router[@id='BorderRouterB']/DA")
**.InteriorRouter.difAllocator.configData =
xmldoc("config.xml",
"Configuration/Router[@id='InteriorRouter']/DA")
**.HostB.difAllocator.directory.configData =
xmldoc("config.xml",
"Configuration/Host[@id='HostA']/DA")
**.BorderRouterA.difAllocator.directory.configData =
xmldoc("config.xml",
"Configuration/Host[@id='HostA']/DA")
**.BorderRouterB.difAllocator.directory.configData =
xmldoc("config.xml",
"Configuration/Host[@id='HostA']/DA")
**.InteriorRouter.difAllocator.directory.configData =
xmldoc("config.xml",
"Configuration/Host[@id='HostA']/DA")

#Enrollment settings
**.InteriorRouter.**.enrollment.isSelfEnrolled = true
**.BorderRouterA.relayIpc.**.enrollment.isSelfEnrolled =
true
**.BorderRouterA.ipcProcess1.**.enrollment.isSelfEnrolled
= true
**.BorderRouterB.ipcProcess1.**.enrollment.isSelfEnrolled
= true
**.BorderRouterA.bottomIpc.enrollment.configData =
xmldoc("config.xml",
"Configuration/Router[@id='BorderRouterA']/Enrollment[@id
='bottomIpc']")
**.BorderRouterA.ipcProcess2.enrollment.configData =
xmldoc("config.xml",
"Configuration/Router[@id='BorderRouterA']/Enrollment[@id
='ipcProcess2']")
**.BorderRouterB.relayIpc.enrollment.configData =
xmldoc("config.xml",
"Configuration/Router[@id='BorderRouterB']/Enrollment[@id
='relayIpc']")
**.HostB.ipcProcess1.enrollment.configData =
xmldoc("config.xml",
"Configuration/Host[@id='HostB']/Enrollment")

#QoS Cube sets
**.ra.qoscubesData = xmldoc("config.xml",
"Configuration/QoS Cubes Set")

[Config Ping]
#PingApp setup
**.forceOrder = true
**.HostA.applicationProcess1.applicationEntity.iae.dstApN
ame = "DestinationB"
**.HostA.applicationProcess1.applicationEntity.iae.dstAeN
ame = "MyPing"
**.HostA.applicationProcess1.applicationEntity.iae.startA
t = 10s
**.HostA.applicationProcess1.applicationEntity.iae.pingAt
= 15s
**.HostA.applicationProcess1.applicationEntity.iae.rate =
5
**.HostA.applicationProcess1.applicationEntity.iae.stopAt
= 20s
**.HostA.applicationProcess1.applicationEntity.iae.size =
1024B
```

8.6.2 config.xml

```

<?xml version="1.0"?>
<Configuration>
  <Host id="HostA">
    <DA>
      <Directory>
        <APN apn="SourceA">
          <DIF difName="TopLayer" ipcAddress="hA" />
        </APN>
        <APN apn="DestinationB">
          <DIF difName="TopLayer" ipcAddress="hB" />
        </APN>
        <APN apn="hA_TopLayer">
          <DIF difName="MediumLayerA" ipcAddress="ha" />
        </APN>
        <APN apn="hB_TopLayer">
          <DIF difName="MediumLayerB" ipcAddress="hb" />
        </APN>
        <APN apn="rA_TopLayer">
          <DIF difName="MediumLayerA" ipcAddress="ra" />
          <DIF difName="MediumLayerAB" ipcAddress="rA" />
        </APN>
        <APN apn="rB_TopLayer">
          <DIF difName="MediumLayerB" ipcAddress="rb" />
          <DIF difName="MediumLayerAB" ipcAddress="rB" />
        </APN>
        <APN apn="rA_MediumLayerAB">
          <DIF difName="BottomLayerA" ipcAddress="ra" />
        </APN>
        <APN apn="rB_MediumLayerAB">
          <DIF difName="BottomLayerB" ipcAddress="rb" />
        </APN>
        <APN apn="rC_MediumLayerAB">
          <DIF difName="BottomLayerA" ipcAddress="rc" />
          <DIF difName="BottomLayerB" ipcAddress="rc" />
        </APN>
      </Directory>
      <NeighborTable>
        <APN apn="hA_TopLayer">
          <Neighbor apn="rA_TopLayer" />
        </APN>
        <APN apn="hB_TopLayer">
          <Neighbor apn="rA_TopLayer" />
        </APN>
      </NeighborTable>
    </DA>
  </Host>
  <Host id="HostB">
    <DA>
      <NeighborTable>
        <APN apn="hA_TopLayer">
          <Neighbor apn="rB_TopLayer" />
        </APN>
        <APN apn="hB_TopLayer">
          <Neighbor apn="rB_TopLayer" />
        </APN>
      </NeighborTable>
    </DA>
    <Enrollment>
      <Preenrollment>
        <SimTime t="5">
          <Connect src="hB_TopLayer" dst="rB_TopLayer" />
        </SimTime>
      </Preenrollment>
    </Enrollment>
  </Host>
  <Router id="BorderRouterA">
    <DA>
      <NeighborTable>
        <APN apn="hB_TopLayer">
          <Neighbor apn="rB_TopLayer" />
        </APN>
        <APN apn="rB_MediumLayerAB">
          <Neighbor apn="rC_MediumLayerAB" />
        </APN>
      </NeighborTable>
    </DA>
    <Enrollment id='bottomIpc'>
      <Preenrollment>
        <SimTime t="1">
          <Connect src="ra BottomLayerA "
dst="rc BottomLayerA" />
        </SimTime>
      </Preenrollment>
    </Enrollment>
  </Router>
  <Router id="BorderRouterB">
    <DA>
      <NeighborTable>
        <APN apn="hA_TopLayer">
          <Neighbor apn="rA_TopLayer" />
        </APN>
        <APN apn="rA_MediumLayerAB">
          <Neighbor apn="rC_MediumLayerAB" />
        </APN>
      </NeighborTable>
    </DA>
    <Enrollment id='relayIpc'>
      <Preenrollment>
        <SimTime t="2">
          <Connect src="rB_TopLayer" dst="rA_TopLayer" />
        </SimTime>
      </Preenrollment>
    </Enrollment>
  </Router>
  <Router id="InteriorRouter">
    <DA>
      <NeighborTable>
        <APN apn="hA_TopLayer">
          <Neighbor apn="rB_TopLayer" />
        </APN>
      </NeighborTable>
    </DA>
  </Router>
  <QoSConfigSet>
    <QoSConfig id="QoSConfig-UNRELIABLE">
      <AverageBandwidth>1200000</AverageBandwidth>
      <AverageSDUBandwidth>1000</AverageSDUBandwidth>
      <PeakBandDuration>2400000</PeakBandDuration>
      <PeakSDUBandDuration>2000</PeakSDUBandDuration>
      <BurstPeriod>10000000</BurstPeriod>
      <BurstDuration>1000000</BurstDuration>
      <UndetectedBitError>0.01</UndetectedBitError>
      <PDUDropProbability>0</PDUDropProbability>
      <MaxSDUSize>1500</MaxSDUSize>
      <PartialDelivery>0</PartialDelivery>
      <IncompleteDelivery>0</IncompleteDelivery>
      <ForceOrder>0</ForceOrder>
      <MaxAllowableGap>0</MaxAllowableGap>
      <Delay>1000000</Delay>
      <Jitter>500000</Jitter>
      <CostTime>0</CostTime>
      <CostBits>0</CostBits>
      <ATime>0</ATime>
    </QoSConfig>
    <QoSConfig id="QoSConfig-RELIABLE">
      <AverageBandwidth>1200000</AverageBandwidth>
      <AverageSDUBandwidth>1000</AverageSDUBandwidth>
      <PeakBandDuration>2400000</PeakBandDuration>
      <PeakSDUBandDuration>2000</PeakSDUBandDuration>
      <BurstPeriod>10000000</BurstPeriod>
      <BurstDuration>1000000</BurstDuration>
      <UndetectedBitError>0.01</UndetectedBitError>
      <PDUDropProbability>0</PDUDropProbability>
      <MaxSDUSize>1500</MaxSDUSize>
      <PartialDelivery>0</PartialDelivery>
      <IncompleteDelivery>0</IncompleteDelivery>
      <ForceOrder>1</ForceOrder>
      <MaxAllowableGap>0</MaxAllowableGap>
      <Delay>1000000</Delay>
      <Jitter>500000</Jitter>
      <CostTime>0</CostTime>
      <CostBits>0</CostBits>
      <ATime>0</ATime>
    </QoSConfig>
  </QoSConfigSet>
</Configuration>

```

8.6.3 EnrollmentStateTable Contents Progress

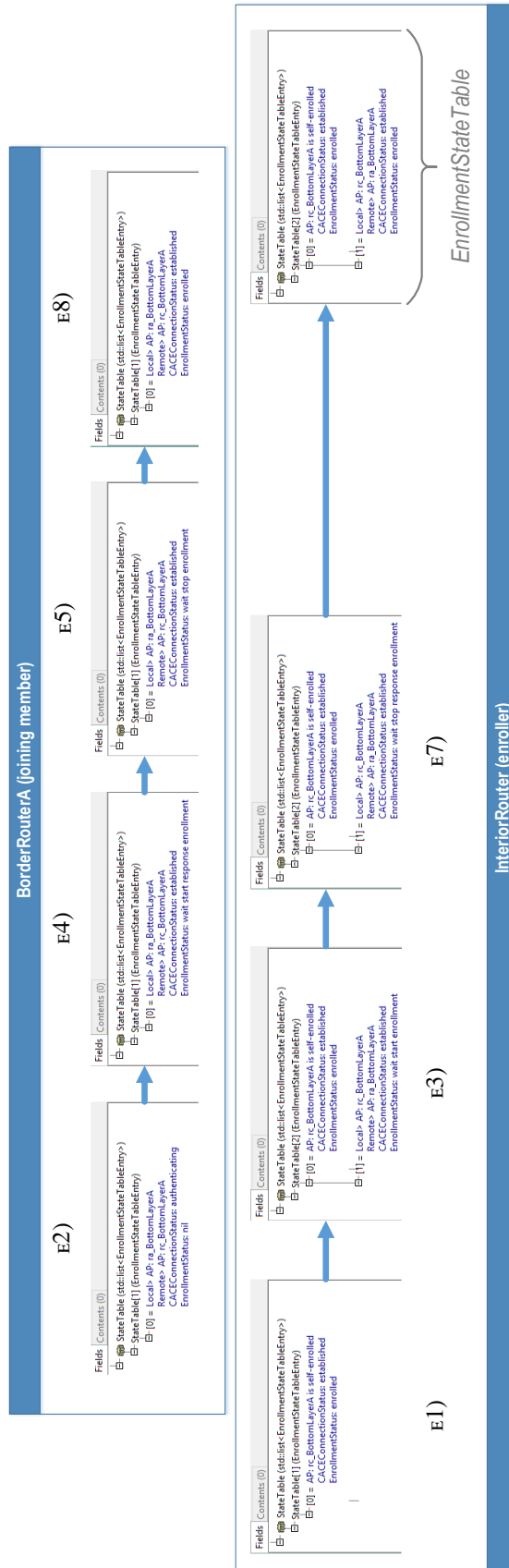


Fig. 83: Content of BottomLayerA's EnrollmentStateTables of BorderRouterA and InteriorRouter

8.6.4 NFlowTable Contents Progress

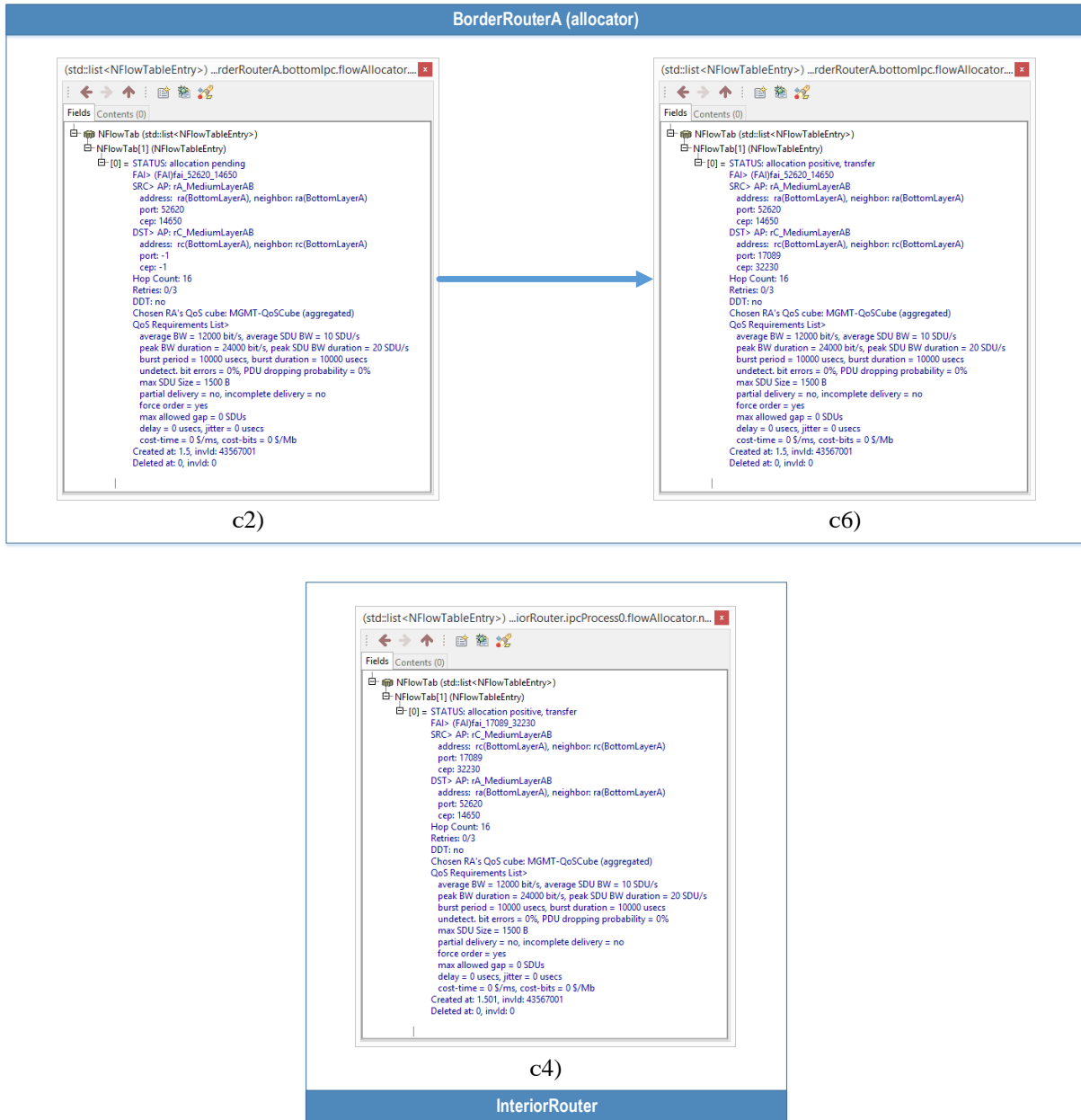


Fig. 84: Content of BottomLayerA's NFlowTables of BorderRouterA and InteriorRouter

9 Lists

9.1 Tables

Tab. 1: Mechanisms related to enrollment phase	5
Tab. 2: Mechanisms related to data transfer	7
Tab. 3: Mechanisms related to control of data transfer	8
Tab. 4: Observations about DFZ based on BGP functionality	21
Tab. 5: Example of relationship between address and name in PSTN	31
Tab. 6: IPv6 address types.....	39
Tab. 7: Design goal importance for a new routing architecture	43
Tab. 8: Properties comparison of existing proposals	54
Tab. 9: Count of map-cache misses under different configurations in scenario with one outage	90
Tab. 10: Count of map-cache misses under different configurations in scenario with two outages	92
Tab. 11: xTR1's statistics for different map-cache synchronization scenarios	94
Tab. 12: xTR_Responder1's statistics for different RLOC probing algorithm scenarios	95
Tab. 13: CDAP message types	117
Tab. 14: Implemented RINASim policies	166

9.2 Figures

Fig. 1: Data transfer mechanisms ontology	9
Fig. 2: IPv4 – All BGP entries in FIB	11
Fig. 3: IPv4 – Active BGP entries in RIB	12
Fig. 4: IPv6 – All BGP entries in FIB	12
Fig. 5: IPv6 – Active BGP entries in RIB	12
Fig. 6: Network multihoming use-case illustration employing simplified BGP rerouting.....	16
Fig. 7: Percentil of IPv4 more specific prefixes	22
Fig. 8: Percentil of IPv6 more specific prefixes	22
Fig. 9: IPv4 FIB table updates	24
Fig. 10: IPv6 FIB table updates	24
Fig. 11: Examples of topologies and non-topologies.....	29
Fig. 12: Homeomorphism illustration.....	29
Fig. 13: Theoretical naming and addressing model for computer networks	36
Fig. 14: Broken Internet naming and addressing model	37
Fig. 15: Fully qualified domain name example and syntax.....	40
Fig. 16: Uniform Resource Identifier structure and examples.....	41
Fig. 17: Core-Edge Separation solution	44
Fig. 18: Core-Edge Elimination solution	45
Fig. 19: CES kinds.....	55
Fig. 20: CEE kinds.....	56
Fig. 21: Basic LISP scheme	60
Fig. 22: LISP packet variants	60
Fig. 23: Comparison between DNS and LISP mapping system.....	63
Fig. 24: LISP-ALT infrastructure example	64
Fig. 25: LISP-DDT infrastructure example.....	65
Fig. 26: LISP-DHT infrastructure example.....	66
Fig. 27: Illustrative LISP unicast data transfer	69
Fig. 28: Illustrative LISP registration process	70
Fig. 29: Illustrative LISP mapping request	71
Fig. 30: Illustrative LISP mapping reply.....	72
Fig. 31: Illustrative communication between LISP and non-LISP world using PITR and PETR	73
Fig. 32: FIT-BUT’s LISP BetaNetwork registration	75
Fig. 33: Site-Based State Synchronization Problem illustration.....	78
Fig. 34: LISP CacheSync message format	81
Fig. 35: LISP CacheSync Acknowledgment message format	81

Fig. 36: Map-cache synchronization operation	82
Fig. 37: Locator Path Liveness Problem illustration.....	83
Fig. 38: LISPRouting module structure	85
Fig. 39: LISP illustrative scenario	85
Fig. 40: Content of xTR_A1's LISPMAPDatabase	87
Fig. 41: Content of xTR_A1's LISPMAPCache	87
Fig. 42: Content of MRMS's LISPSiteDatabase.....	88
Fig. 43: LISP testing network for Map-Cache synchronization	89
Fig. 44: xTR1's LISP control messages occurrence and total processed byte size in scenario with single outage.....	91
Fig. 45: xTR1's LISP control messages occurrence and total processed byte size in scenario with two outages	92
Fig. 46: xTR1's LISP control messages occurrence and total processed byte size in scenario with two outages + ack.....	93
Fig. 47: LISP testing network for merged RLOC probing.....	94
Fig. 48: xTR_Responder1's LISP messages occurrence and total processed byte size in scenario with forty EIDs.....	96
Fig. 49: Application Protocol and Application Entities relationship	99
Fig. 50: DIF, DAF, DAP and IPCP illustration	101
Fig. 51: IPCP local identifiers overview	104
Fig. 52: Example of RINA network with three levels of DIFs and different nodes	105
Fig. 53: Distributed Application Process components	106
Fig. 54: IPC Process components	107
Fig. 55: Message passing between RINA components	108
Fig. 56: EFCP instance divided into DTP and DTCP part.....	109
Fig. 57: Flow allocation process	112
Fig. 58: Flow Allocator operation.....	113
Fig. 59: Flow Allocator Instance operation of initiating IPCP.....	114
Fig. 60: Flow Allocator Instance operation of responding IPCP before the flow was allocated.....	115
Fig. 61: Flow Allocator Instance operation after the flow was allocated.....	115
Fig. 62: Establishment phase on initiating process	118
Fig. 63: Establishment phase on responding process.....	119
Fig. 64: Data transfer phase on initiating/responding process.....	119
Fig. 65: Host nodes structure examples	122
Fig. 66: Router nodes structure examples	123
Fig. 67: DAF components for RINASim	125
Fig. 68: IPCP's DIF components for RINASim.....	128

Fig. 69: RINASim demonstration topology	130
Fig. 70: Visualization RA's available QoS-cubes	132
Fig. 71: Visualization of Directory mappings	132
Fig. 72: Data transfer phase illustration	139
Fig. 73: Content of TopLayer ipcProcess1 NFlowTables for HostA and HostB	140
Fig. 74: LISP Map-Request message format.....	158
Fig. 75: LISP Map-Reply message format.....	159
Fig. 76: LISP Map-Register message format	160
Fig. 77: LISP Map-Notify message format	160
Fig. 78: ANSARouter module structure.....	161
Fig. 79: xTR1's LISP control messages occurrence and total processed byte size in scenario with single outages + ack.....	162
Fig. 80: xTR1's LISP control messages occurrence and total processed byte size in scenario with three outages	162
Fig. 81: xTR1's LISP control messages occurrence and total processed byte size in scenario with two outages + ack.....	163
Fig. 82: xTR_Responder1's LISP messages occurrence and total processed byte size in scenario with eighty EIDs.....	163
Fig. 83: Content of BottomLayerA's EnrollmentStateTables of BorderRouterA and InteriorRouter .	169
Fig. 84: Content of BottomLayerA's NFlowTables of BorderRouterA and InteriorRouter	170

9.3 Index

A

A Practical Transit-Mapping Service (APT)	51
access control	5
access-lists (ACLs)	19
acknowledgement (ack)	7
address	27, 30
address overloading	14
address space	30
addressing	6
advertisement timer (AT)	77
Aggregated Congestion Control (ACC)	144
AllocateNotifyPolicy	113
AllocateRetryPolicy	113
Alternative Topology (LISP-ALT)	64
ANSA project	84
ANSARouter	84
anti-route hijacking	20
Application Entity (AE)	99
Application Entity Instance Identifier (AEI-id) ..	103
Application Entity Name (AEN)	103
Application Naming Information (ANI)	103
Application Process (AP)	99
Application Process Instance Identifier (API-id)	103
Application Process Name (APN)	103
application protocols	99
architecture	3
assignment	28
association	4
authentication	5
autonomous system number (ASN)	16
autonomous systems (AS)	13
AVM Fritz!OS	74

B

Backup	77
binding	4, 28

bit stuffing	6
Border Gateway Protocol (BGP)	11
Border routers	105
bound	28
BSD sockets	51
business acquisitions	20

C

combination	6
Common Distributed Application Protocol (CDAP)	116
compression	7
confidentiality	7
config.xml	131
congestion control	8
connection	4, 103
Connection-endpoint-id (CEP-id)	103
Connection-id	103
control plane	13
Core-Edge Elimination (CEE)	44
Core-Edge Separation (CES)	44

D

Data Anti-Corruption	7
Data Transfer Control Protocol (DTCP)	109
Data Transfer Phase	5
Data Transfer Protocol (DTP)	109
Data transfer protocols	99
deassignment	28
Default Free Zone (DFZ)	11
Delegated Distributed Tree (LISP-DDT)	65
delimiting	6
Delimiting	109
Delta-t	102
device	1
DFZ RIB/FIB growth	13
DIF Allocator (DA)	106

direct alias	28
Directory	107
Distributed Application Facility (DAF)	100
Distributed Application Name (DAN)	103
Distributed Application Process (DAP)	100
Distributed Hash Tables (LISP-DHT)	65
Distributed IPC Facility (DIF).....	100
Domain Name System (DNS)	40
dual-stack.....	20

E

EFCP instance (EFCPI)	109
Egress Tunnel Router (ETR).....	61
Endpoint Identifier (EID).....	59
end-site renumbering	20
end-to-end principle.....	17
Enrollment.....	108
Enrollment Phase.....	5
Error and Flow Control Protocol (EFCP).....	109
error correction	7
error detection	7
Establishment Phase	5
Evolution.....	50
Exterior Gateway Protocol (EGP)	18

F

Flag Day.....	20
flow.....	4, 103
Flow Allocator (FA)	111
Flow Allocator Instance (FAI)	111
Flow object.....	111
forwarding.....	6
Forwarding Information Base (FIB)	11
fragmentation	6
fully qualified domain name (FQDN).....	40

G

Global Locator, Local Locator, and Identifier Split (GLI-Split)	48
--	----

H

Hierarchical IPv4 Framework (hIPv4)	47
homeomorphism	29
Host Identifier Protocol (HIP)	45

I

identification.....	14
Identifier.....	14
Identifier-Locator Network Protocol (ILNP).....	49
inbound traffic engineering	18
indirect alias	28
Ingress Tunnel Router (ITR)	61
Initial State Synchronization.....	7
interconnection richness	23
Interior Gateway Protocol (IGP).....	18
Interior routers	105
Internet Architecture Board (IAB).....	1
Internet Corporation for Assigned Names and Numbers (ICANN)	10
Internet of Things (IoT).....	17
Internet Service Provider (ISP).....	10
Internet Vastly Improved Plumbing (Ivip)	47
inter-process communication (IPC)	100
IP hijacking.....	24
IPC API.....	108
IPC Management	106
IPC Process (IPCP).....	100
IPC Resource Manager (IRM).....	107
IPv4 address exhaustion	20
IRATI.....	120
IRINA	120

K

keepalives	6
------------------	---

L

layer	4
Level 3 Multihoming Shim Protocol for IPv6 (Shim6)	46

LISP BetaNetwork.....	75
LISP CacheSync.....	80
LISP CacheSync Ack.....	80
LISP Internet Groper (LIG).....	75
LISP Map-Notify.....	63
LISP Map-Referral.....	65
LISP Map-Register.....	63
LISP Map-Reply.....	63
LISP Map-Request.....	63
LISP mobile node.....	74
LISP Negative Map-Reply.....	63
LISPmob.....	74
LIST Network Address Translation (LISP-NAT)	67
load-balancing.....	15
Local Internet Registries (LIR).....	10
localization.....	14
locate.....	30
location dependent.....	30
locator.....	14
Locator Path Liveness Problem.....	76
Locator/ID Split Protocol (LISP).....	1
loosely-bound.....	102
lost and duplicity detection.....	7

M

management flows.....	136
map cache.....	61
Map Resolver (MR).....	62
Map Server (MS).....	62
map-and-encap.....	43
mapping database.....	63
master.....	77
Master down interval (MDI).....	77
Maximum Packet Lifetime (MPL).....	102
mechanism.....	5
merged RLOC probing.....	84
mobility.....	17
more specific prefixes.....	20
multihoming.....	15
multiplexing.....	6

Multiprotocol Label Switching (MPLS).....	59
---	----

N

naïve synchronization.....	79
name.....	27
Name Overlay Service for Scalable Internet Routing (NOL).....	48
name space.....	27
Name-Base Sockets (NBS).....	51
Namespace Management (NSM).....	111
Naming information table.....	106
National research and education network (NREN)	120
Neighbor table.....	106
Network Address Translation (NAT).....	14
network interface cards (NIC).....	106
NewFlowRequstPolicy.....	113
node.....	1
non-broadcast multi-access (NBMA).....	52
nonLISP.....	59
nonrepudiation.....	7

O

object.....	27
object sharing.....	32
omnetpp.ini.....	131
OpenLISP.....	74
ordering.....	6
organization.....	18
OSI Reference Model (OSI-RM).....	3
outbound traffic engineering.....	18

P

pathname.....	33
PDU Forwarding Table.....	116
PDU Forwarding Table Generator.....	116
perform routing.....	35
Point of Attachment (PoA).....	14
Point-to-Point Tunneling Protocol (PPTP).....	59
policy.....	5

portability.....	17
Port-id.....	103
PRISTINE.....	120
protocol.....	4
protocol control information (PCI).....	4
protocol data unit (PDU).....	4
protocol machine (PM).....	4
ProtoRINA.....	120
Provider Aggregatable (PA).....	10
Provider Independent (PI).....	10
Proxy Egress Tunnel Router (PETR).....	67
Proxy Ingress Tunnel Router (PITR).....	67
Pull model.....	62
Push model.....	62
PxTR.....	67

Q

Quality of Service (QoS).....	5
-------------------------------	---

R

rank.....	4
rapid shuffling of prefixes.....	23
reassembling.....	6
Recursive Internet Architecture (RINA).....	1
redundancy.....	15
Regional Internet Registry (RIR).....	18
Rekhter's Law.....	14
relaying.....	6
Relaying and Multiplexing Task (RMT).....	110
renumbering.....	18
Resource Allocator (RA).....	116
Resource Information Base (RIB).....	116
retransmission.....	8
reverse path forwarding (RPF) check.....	50
rewriting.....	43
RIB Daemon (RIBd).....	116
RINA Simulator (RINASim).....	121
RIR allocation policies.....	20
RLOC probing.....	64
route.....	27, 30

route dependent.....	30
router.....	10
routing.....	6
Routing Architecture for the Next Generation	
Internet (RANGI).....	46
Routing Information Base (RIB).....	11
Routing Locator (RLOC).....	59
Routing Research Group (RRG).....	1
routing scalability.....	13
R-timer.....	102

S

saturation point.....	21
Scalable Forwarding with RINA (SFR).....	144
scope.....	4
SDU Protection.....	110
Search table.....	106
segmentation.....	6
SeqRollOverPolicy.....	113
sequence numbers.....	6
Service data unit (SDU).....	4
Service-level Agreements (SLA).....	18
simple merged RLOC probing.....	84
Site-Based State Synchronization Problem.....	78
Site-Based Synchronization Problem.....	76
Six/One Router.....	52
smart synchronization.....	80
Solicit-Map-Request (SMR).....	82
sophisticated merged RLOC probing.....	84
state vector.....	102
string over alphabet.....	27
Subnetwork Encapsulation and Adaptation Layer (SEAL).....	52

T

three-way handshake.....	4
tightly-bound.....	102
time to live (TTL).....	50
time-to-live (TTL).....	79
top-level domain.....	40

topological space	29
topologically dependent	30
topology	29
traffic engineering (TE)	18
Tunneled Inter-Domain Routing (TIDR)	49
Tunneling Route Reduction Protocol (TRRP).....	52
two-way handshake	4

U

unambiguous	28
Uniform Resource Identifier (URI)	41
Uniform Resource Locator (URL).....	41
Uniform Resource Name (URN).....	41
unique	28

URI authority.....	41
URI scheme	41

V

Virtual Private Networks (VPNs)	100
Virtual Router ID (VRID)	77
Virtual Router Redundancy Protocol (VRRP).....	76
VRRP priority.....	77

W

World Wide Web (WWW).....	41
---------------------------	----

X

xTR	61
-----------	----