VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

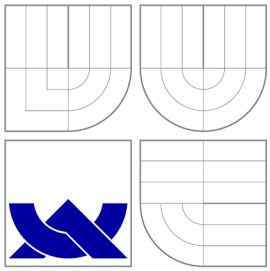
MODELOVÁNÍ PROTOKOLŮ IS-IS A TRILL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

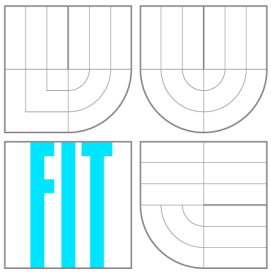
AUTOR PRÁCE
AUTHOR

Bc. MARCEL MAREK

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ PROTOKOLŮ IS-IS A TRILL

MODELLING IS-IS AND TRILL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARCEL MAREK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR VESELÝ

BRNO 2013

Abstrakt

V této práci jsou popsány principy směrovacího protokolu IS-IS. Je představen aktuální stav implementace tohoto protokolu v rámci simulačního frameworku OMNeT++. Dále je ukázána implementace protokolu IS-IS vytvořená v rámci projektu ANSA. Navíc je vytvořena jeho varianta nazývaná TRILL, která je v současné době nasazována jako náhrada za STP v prostředí datových center. Cílem práce je umožnit modelování daných protokolů bez nutnosti budování fyzické architektury.

Abstract

In this thesis, we describe the principles of IS-IS routing protocol. We introduce the current state of implementation of this protocol within the simulation framework OMNeT++. We present the implementation of the IS-IS protocol created within the ANSA project. Moreover, we employ its variant called TRILL that is nowadays deployed as replacement of STP in data-center environment. The aim is to enable the modelling of the protocols without the need having to build physical architecture.

Klíčová slova

IS-IS, ANSA, OMNeT++, INET, směrovací protokol, TRILL, RBridge, linková vrstva, počítačové sítě

Keywords

IS-IS, ANSA, OMNeT++, INET, routing protocol, TRILL, RBridge, data link layer, computer networks

Citace

Marcel Marek: Modelování protokolů IS-IS a TRILL, diplomová práce, Brno, FIT VUT v Brně, 2013

Modelování protokolů IS-IS a TRILL

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Veselého.

.....
Marcel Marek
22. května 2013

Poděkování

Na tomto místě bych chtěl poděkovat své mamince za podporu během celého studia a především za chystání výborných svačin, které mi umožnily plně se koncentrovat na přednáškách.

Rád bych poděkoval také svému vedoucímu Ing. Vladimíru Veselému za jeho vstřícný přístup, ochotu, věcné připomínky a humor, kterým mě provázel od prvního semestru magisterského studia. Jako poděkování bych mu rád věnoval recept na kulinářskou pochoutku o dvou chodech nazvanou „Květákovo dobrodružství“.

Na první chod *Zuppa di cavolfiore* budeme potřebovat následující suroviny: jeden mladý květák, cibuli, 500ml zeleninového vývaru, 50ml smetany, pepř, sůl, slaninu a parmazán. Při přípravě postupujeme metodou *bottom-up*. Cibuli opečeme do zlatova, přidáme květák a 3 minuty opekáme. Zalijeme zeleninovým vývarem a zvolna vaříme 15 minut. Výsledek rozmixujeme a přidáme smetanu. Metodou *shoot from the hip* dochutíme výsledný produkt solí a pepřem. Dozdobíme opečenou slaninou a parmazánem.

Jako druhý chod podáváme *Crepes coliflor*.
Nachystáme si květák, 2 vejce, mouku a mléko. Přípravu zahájíme přístupem *divide and conquer* s vejci. Nastrouháme květák a přidáme mléko. Přidáme žloutky a zahustíme moukou do konzistence těsta. Z bílků ušleháme sníh a vmícháme do těsta. Vytvoříme placičky a smažíme na pánvi z obou stran. Pro přípravu přílohy kontaktujte Vašeho administrátora. Oba chody servírujeme se zvoláním: „A jez, a jez!“.

Obsah

Kapitola 1

Úvod

V dnešní době moderních technologií stále více stoupá potřeba být neustále s někým v kontaktu. Ať už se jedná o práci, rodinu či zábavu, všichni dnes chtějí být co nejvíce času *online*.

Z tohoto důvodu je nutné nasadit takové směrovací protokoly, které toto budou schopné co možná nejlépe zajistit. Mezi hlavní faktory, které dnes rozhodují jaký směrovací protokol nasadit na nově vznikající síti, jsou jeho otevřenost a možnost jej bezpečně a spolehlivě otestovat před samotným nasazením.

V datových centrech, která jsou kritická pro fungování a provozování webových služeb jako Google a Wikipedia je důležité mít kvalitní protokoly nižších vrstev.

Odpovědí na výše zmíněné problémy a požadavky může být IS-IS. Je to otevřený směrovací protokol a existuje pro něj rozšíření pro podporu protokolu TRILL, který zajišťuje funkcionalitu na linkové vrstvě.

Pro jednoduché, bezpečné a spolehlivé testování můžeme využít simulačního prostředí OMNeT++. Nejprve ale musíme dané protokoly implementovat a tomu se budeme věnovat v této práci.

1.1 Cíle práce

Cílem práce je poskytnout možnost jednoduše a bezpečně simulovat protokol IS-IS a TRILL v prostředí OMNeT++. Abychom mohli tento cíl splnit, musíme rozšířit stávající knihovny pro OMNeT++ umožňující simulování počítačových sítí, INET a ANSAINET.

Zjistíme tedy aktuální stav podpory zmíněných technologií v OMNeT++. Kromě vytvoření implementace samotných protokolů, přidáme také podporu pro CLNS směrování formou nové směrovací tabulky.

Vytvořenou implementaci ověříme porovnáním s reálnou sítí. K tomu využijeme zařízení firmy Cisco, která jsou dostupná ve školní laboratoři.

1.2 Struktura práce

Ve druhé kapitole si projdeme aspekty směrovacího protokolu IS-IS. Představíme si používané formáty zpráv a způsob jejich výměny na různých topologiích. Podrobně si popíšeme práci s link-state databází, ukážeme si jakým způsobem jsou jednotlivé LSP označovány, jak je zaručena aktuálnost a expirace starých LSP a jak probíhá synchronizace mezi jednotlivými IS. Ukážeme důležitou roli DIS na broadcast rozhraních. A zmíníme také způsob

konfigurace protokolu IS-IS na Cisco zařízeních.

Třetí kapitola se věnuje protokolu TRILL. Probereme si jakým způsobem využívá ke své činnosti protokol IS-IS. Představíme zařízení v rámci kterého je protokol definován a principy přeposílání.

V čtvrté kapitole uvádíme hlavní principy protokolu IS-IS a TRILL tak, jak byly implementovány v rámci projektu ANSA v simulačním prostředí OMNeT++. Zmiňujeme odchylky od návrhu společně s odůvodněním dané změny. Podrobněji popisujeme způsob konfigurace. Seznámíme se také se způsobem plánování zpráv a jejich následné generování a odesílání. Součástí je i popis rozšíření IS-IS pro podporu protokolu TRILL a TRILL samotný. Kromě těchto protokolů je také popsán

Pátá kapitola je věnována testování. Na několika příkladech porovnáme chování modelovaného protokolu IS-IS s implementací na reálných zařízeních. Protože aktuálně není zařízení, které by podporovalo protokol TRILL, ukážeme si na jednoduchém příkladu výhody oproti protokolu STP.

Poslední kapitola shrnuje dosud dokončenou práci v teoretické části i v rámci implementace. Upozorňuje na nedostatky aktuálního stavu a navrhuje možná řešení.

Kapitola 2

IS-IS

V této kapitole se seznámíme s protokolem IS-IS. Představíme si používané formáty zpráv a způsob jejich výměny na různých topologiích.

IS-IS celým názvem *Intermediate System to Intermediate System* je dvou úrovňový směrovací protokol běžící na síťové vrstvě modelu ISO/OSI. Každý prvek v sítí označuje jako *System*. *End System* označuje *koncový* prvek, za který můžeme považovat například PC. Naopak, zařízení podílející se na doručování dat, jsou označovány jako *Intermediate System* (IS). Triviálně více používaný termín pro *Intermediate System* je *router*.

IS se sdružují do oblastí (*area*). Směrování v rámci jedné oblasti označujeme jako *L1 směrování*. Naopak směrování mezi oblastmi jako *L2 směrování*.

V IS-IS rozlišujeme pouze dva typy rozhraní, broadcastové a point-to-point.

Na broadcastových rozhraních jako je Ethernet se nepoužívají unicastové adresy, ale pouze multicastové. IS-IS se snaží, aby každý systém připojený do dané LAN slyšel každou zprávu. Ke komunikaci se všemi *L1 IS* využívá multicastovou MAC adresu `0180:c200:0014` a pro *L2 IS* adresu `0180:c200:0015`. Jako zdroj se většinou používá adresa daného Ethernet portu, přes který byla zpráva odeslána. Dále je v ethernetové hlavičce uveden DSAP a SSAP kód 0xFE určující, že se jedná o OSI protokol na obou stranách spojení. Uvnitř takového rámce se nachází nativní IS-IS zpráva, která může být v intervalu $< 27B, MTU - 21B >$, kde MTU značí velikost MTU dané linky a $21\,B$ je součet ostatních částí Ethernetového rámce. IS-IS proces musí sám zaručit, že nepřekročí tuto velikost, i kdyby to mělo znamenat rozdělení zprávy do několika rámců, protože Ethernet samotný nepodporuje fragmentaci.

Kapitola IS-IS byla zpracována mimo jinde uvedených zdrojů na základě materiálů v [19], [9], [4], [21], [20] a [5].

2.1 Oblasti

OSI používá pro rozdělení rozsáhlé sítě na menší celky pojem oblast (*Area*). Slouží ke zmenšení množství uzlů, mezi kterými je třeba propočítat nejkratší cestu pomocí Dijkstrova algoritmu nazývaného také Shortest Path First (SPF). Ovšem stále potřebujeme způsob jak komunikovat mezi jednotlivými oblastmi. K tomu slouží IS pracující na druhé úrovni (L2 IS), které se starají o výměnu informací o dostupných sítích v jednotlivých oblastech. Nutno podotknout, že každý IS může být součástí více oblastí zároveň. Hranice jednotlivých oblastí nejsou na IS/směrovačích, jak je tomu například u OSPF, ale na linkách, jako např. u BGP.

Způsob jakým můžeme u IS-IS dosáhnout hierarchie spočívá v použití úrovní.

2.2 Úrovně

Jak jsme zmínili na začátku, jedná se dvou úrovňový protokol. Jelikož hranice oblastní neleží na zařízeních, ale na linkách, musíme nějakým způsobem určit jak se budou dané linky chovat při navazování sousedství (*adjacency*). Protože, jak jsme zvyklí například z OSPF, pro navázání sousedství musí být oba routery součástí jedné oblasti. To ovšem u IS-IS platí pouze pro *Level 1* sousedství.

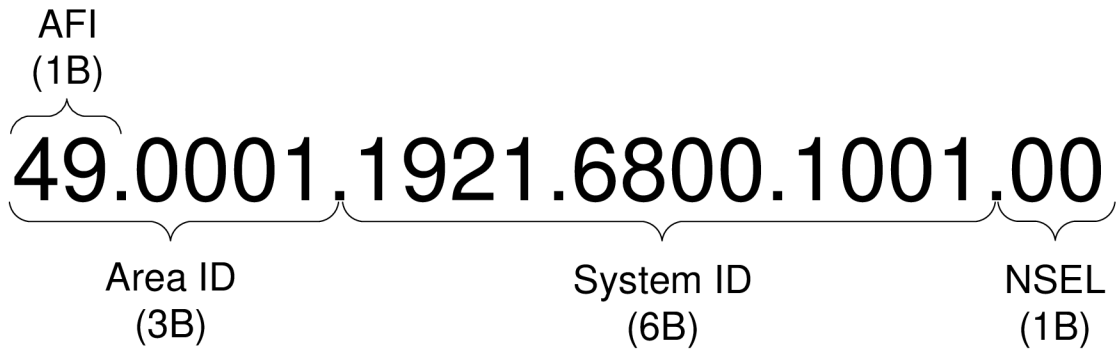
Proto používáme dvě úrovně – *L1* a *L2*[1]. Každý router v IS-IS síti buduje dvě odlišné topologie – *Level 1* a *Level 2*. Každá linka je označena jednou ze tří značek – *L1*, *L2*, nebo *L1L2*, která určuje do jaké hierarchické úrovně chce patřit. Díky tomu můžeme určit, že u *L2* linky není nutná podmínka shody identifikátoru oblasti (*Area-ID*) pro navázaní sousedství. Jedinou podmínkou pro *L2* páteřní spojení je, že musí být souvislé. Žádný *L2 IS* nesmí být izolován od ostatních *L2 IS*. Nicméně pro *L1 IS* je pořád nutná shoda *Area-ID* na obou stranách, pro navázání sousedství stejně jako je tomu u OSPF.

2.3 OSI adresování

IS-IS používá adresní model síťových protokolů OSI. Největší změnou oproti IP adresování je, že IS používá pouze jedinou OSI adresu pro IS, tedy všechna rozhraní. Taková adresa je většinou přiřazena směrovacímu procesu (Cisco IOS), případně virtuálnímu loopback rozhraní (Juniper JUNOS). Můžeme ovšem přiřadit i několik adres – využívá se při migraci oblastí. Nutno podotknout, že všechny adresy platí pro všechna rozhraní daného IS. V OSI adresování se používá mnoho specifických termínů, některé jsme zmínili již na začátku, jiné nikoli. Shrňme si proto ty nejdůležitější.

2.3.1 NET

IS musí mít nakonfigurován alespoň jeden NET (*Network Entity Title*). Jednoznačně identifikuje IS, nikoliv pouze adresu rozhraní jako je tomu v případě IP adresy. NET zahrnuje podmnožinu adresních formátů v OSI adresování. Použitý formát identifikuje první byte NET viz sekce 2.3.4. Strukturu s vyznačenými segmenty můžeme vidět na obrázku 2.1. NET je velký od 3 do 22 B a skládá se ze tří částí:



Obrázek 2.1: Formát Network Enttity Title (NET).

[1] L jako zkratka z anglického Level – úroveň.

2.3.2 NSEL

NSEL, neboli *Network Selector Label*, je poslední byte v NET. Jeho hodnota, pro použití v IS-IS, musí být vždy `0`. V jiném případě by nedošlo k navázání sousedství mezi dvěma IS. NSEL je obdobou identifikátoru protokolu v IP hlavičce a umožňuje provozování několika služeb na daném NET.

2.3.3 System-ID

Každý link-state protokol musí zajistit unikátní identifikaci všech uzlů v síti pro svoje fungování. Dle specifikace v ISO 10589:2002, *System-ID* může být různé délky od 1 do 8 B. Nicméně současné implementace protokolu IS-IS používají ustálenou velikost 6 B. Konkrétní používaná délka je specifikována v IIH (IS-IS Hello) hlavičce v položce *ID Length* velké 1 B. Běžným způsobem jak zajistit unikátnost *System-ID* v síti je použití dekadického zápisu 32 b IP adresy, kde každý byte je zapsán jako trojice cifer. Pokud daná hodnota zabírá v dekadickém zápisu méně než tři cifry, doplní se zleva nulami. Výsledná adresa se pak zapisuje hexadecimálně vždy po dvou bytech oddělených tečkou. Pro IP adresu `192.168.1.1` by *System-ID* vypadalo následovně `1921.6800.1001`. Způsob zajištění unikátnosti není nijak definován a je čistě na rozhodnutí administrátora.

2.3.4 Area-ID

Area-ID může mít od 1 do 13 B. Formát je specifikován v prvním bytu tzv. AFI – *Address Family Identifier*, někdy také označován jako *Authority and Format Identificator*. AFI kódy spravují společně organizace ITU-T a ISO/IEC a jsou uvedené v ISO 8348 [14]. Při definování nového typu, musí být ISO 8348 aktualizováno.

Mezi dobře známé AFI patří například

- 39 DCC (data country code),
- 45 E.164,
- 46 ICD (Internation Code Designator) a
- 49 private addressing.

AFI 49 označující *private-addressing* (*soukromé adresování*) umožňuje používání libovolného formátu a je jakousi obdobou RFC 1918 [23] pro používání privátních adres v IP sítích.

V návaznosti na příklad zápisu v sekci 2.3.3 by celá OSI adresa mohla vypadat takto `49.0001.1921.6800.1001.00`. První byte specifikující *private addressing*, následovaný dvěma byty pro *Area-ID*, 6 B *System-ID* a poslední byte *NSEL*, který musí být vždy nulový.

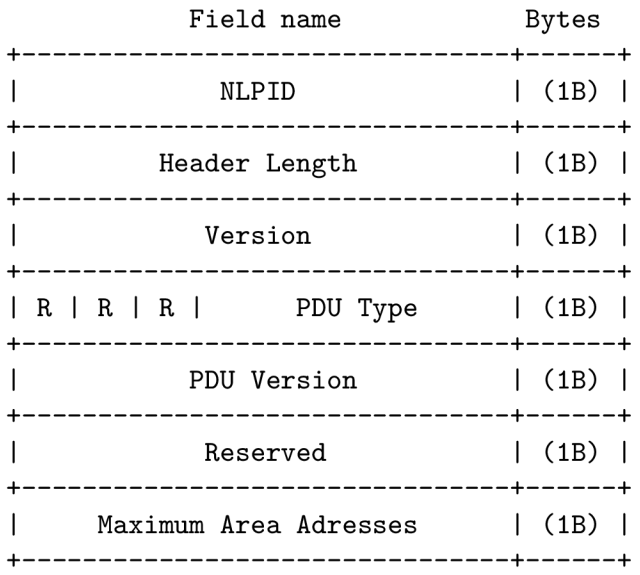
2.4 Typy zpráv

IS-IS používá tři základní typy zpráv. *Hello* zprávy pro objevování sousedů a navazování sousedství. LSP - *Link State PDU* pro výměnu informací o síťové topologii a *Sequence Number PDU* pro synchronizaci LSP databází mezi jednotlivými IS. V této části popíšeme dané zprávy a jejich varianty.

Všechny typy zpráv tvoří tři části: *obecná hlavička*, hlavička specifická pro daný typ zprávy a TLV část.

2.4.1 Obecná hlavička

Obecná hlavička, neboli IS-IS hlavička předchází všem IS-IS zprávám. Její délka je 8 B a říká příjemci o verzi používaného protokolu, délce hlavičky, maximálním počtu provozovaných oblastí, společně s dalšími IS-IS parametry jako délka *System ID*. Formát hlavičky viz obrázek 2.2.



Obrázek 2.2: Formát obecné hlavičky IS-IS

NLPID

V prostředí OSI má každý protokol přidělený 1 B velký kód. Pro IS-IS je to hodnota `0x83`.

Délka hlavičky

Určuje délku hlavičky v bytech včetně hlavičky pro daný typ PDU.

Verze protokolu

Udává momentálně používanou verzi. Aktuální hodnota je `1`.

Délka *System-ID*

Určuje délku *System-ID* viz 2.3.3.

Rezervované bity

Jedná se o 3 b, při odesílání se nastavují na hodnotu `0` a při přijetí jsou ignorovány.

Typ

Typ (*PDU Type*) je 5-ti bitová položka a specifikuje jeden z následujících typů zprávy:

- Level 1 LAN IS to IS Hello PDU
- Level 2 LAN IS to IS Hello PDU
- Point-to-Point IS to IS Hello PDU
- Level 1 Link State PDU
- Level 2 Link State PDU
- Level 1 Complete Sequence Numbers PDU
- Level 2 Complete Sequence Numbers PDU
- Level 1 Partial Sequence Numbers PDU
- Level 2 Partial Sequence Numbers PDU

PDU Version

Označuje verzi formátu výše zmíněného PDU typu. Aktuální hodnota je `1`.

Reserved

Rezervované pole délky 1 B.

Maximální počet oblastí

Maximální počet povolených oblastí, jichž daný IS může být součástí.

2.4.2 Hello zprávy

Směrovací protokoly používají *Hello* zprávy pro objevování sousedů a pro vyjednání parametrů spojení. V případě IS-IS se jedná o *IS-IS Hello* (IIH) zprávy. IS-IS podporuje několik typů IIH, podle toho o jakou úroveň (L1, L2) a o jakou topologii (point-to-point, broadcast) se jedná. Pro snížení šířky používaného pásma pro point-to-point linku v případě provozování L1 i L2 IIH je definováno, že na takové lince používá jeden typ pro obě úrovně.

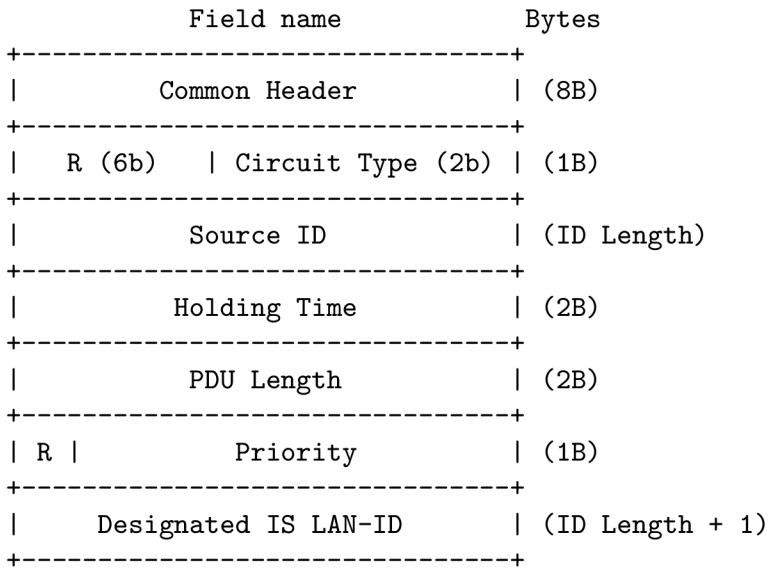
Jako všechny IS-IS zprávy i IIH začínají IS-IS hlavičkou. Jaký typ zprávy následuje za touto hlavičkou určuje *PDU type*. *PDU type* pro IIH může být jeden z trojice:

- 15 – L1 LAN IIH,
- 16 – L2 LAN IIH a
- 17 – point-to-point IIH.

PDU type 17 určuje, že se jedná o IIH pro *L1*, *L2*, nebo *L1L2* úroveň na point-to-point lince.

2.4.3 LAN Hello zprávy

Tento typ zpráv se používá na broadcastových sítích (LAN). Délka *LAN Hello* zprávy je vždy 27 B. Skládá se z běžné hlavičky (8 B) a LAN Hello hlavičky (19 B). *PDU type* je vždy `15`, nebo `16`, podle toho, zda se jedná o *Hello* zprávu pro L1, nebo L2 IS. Strukturu LAN Hello hlavičky vidíme na obrázku 2.3.



Obrázek 2.3: Formát LAN Hello PDU

Reserved

Jedná se o prvních 6 b, které jsou rezervované a nepoužívají se. Jejich hodnota by měla být `0`.

Circuit type

Poslední dva bity prvního bytu určují, jaké úrovně byly nastaveny pro tuto linky. Povolené hodnoty jsou:

- 0x1 pro L1,
- 0x2 pro L2,
- 0x3 pro L1 a L2 současně.

V případě, že se v tomto poli vyskytne jiná hodnota, předpokládáme, že se někde stala chyba a danou Hello zprávu zahodíme.

Source ID

Délka tohoto pole se odvíjí od hodnoty *ID length* z *common header*, nejčastěji tedy 6 B a obsahuje *System-ID* odesílacího IS.

Holding Time

Holding Time říká, po jaké době chce být zdrojový IS prohlášen za nedostupného. Jinými slovy, pokud se IS neozve do daného počtu vteřin, budeme navázané sousedství považovat za zrušené a provedeme potřebné kroky. Takové kroky obvykle obnášejí rozeslání zprávy o zrušení sousedství okolním IS. Každá přijatá *Hello* zpráva vynuluje čítač odpočtu prohlášení za nedostupného na `0`. Na rozdíl např. od protokolu OSPF, se *Hello time* a *Hold time* intervaly nemusí shodovat, aby došlo k navázání sousedství. Každý IS oznamuje svůj interval sousedům a může jej tak případně měnit i v době provozování IS-IS.

PDU Length

PDU Length obsahuje délku celého paketu v B včetně *obecné hlavičky* a *LAN Hello hlavičky*.

Priority

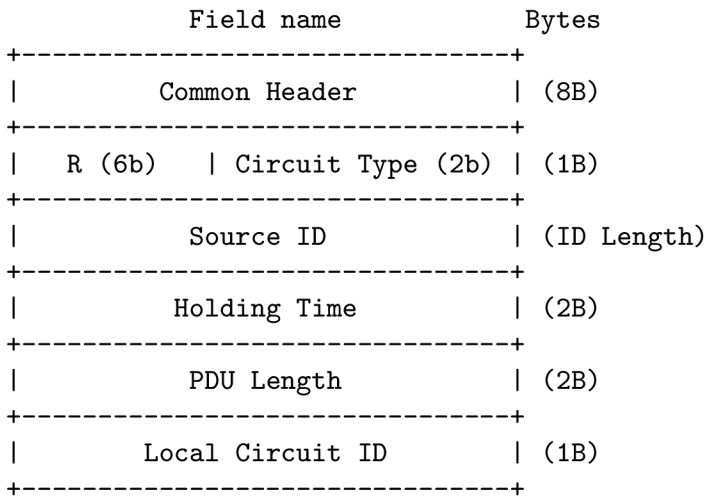
Priority společně s *Designated IS LAN-ID* hrají roli ve volbě designated IS.

Designated IS LAN-ID

Obsahuje *LAN-ID* aktuálně zvoleného DIS z pohledu odesílajícího IS. Při prvotním zasílání jej odesílající IS vyplní svým vlastním.

2.4.4 Point-to-point Hello zprávy

Na rozdíl od LAN Hello zprávy, na point-to-point lince nepotřebujeme *Priority* ani *DIS ID*, protože nedochází k volbě DIS. Formát, viz obrázek 2.4, i význam ostatních polí zůstal stejný jako v LAN Hello.

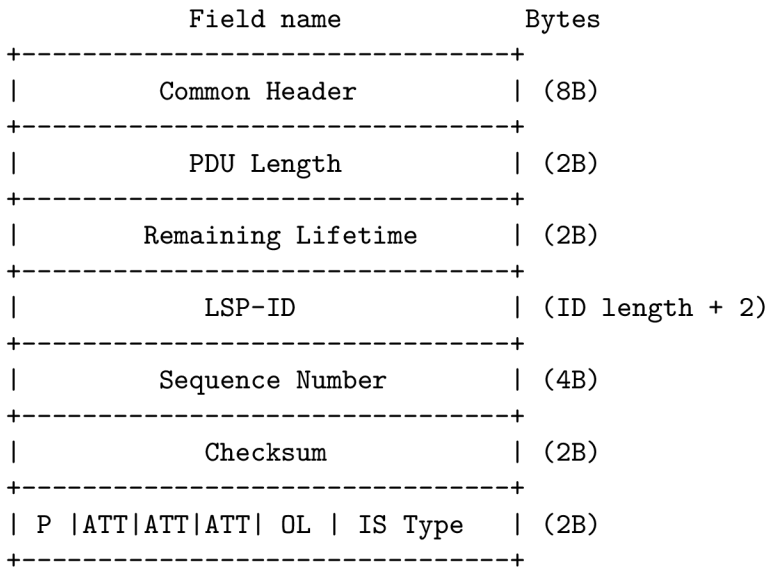


Obrázek 2.4: Formát Point-to-point Hello PDU

Local Circuit-ID je *nepodstatná* část protokolu a nemusí být nastavena. Některé implementace ho nastavují na předem danou konstantu. Informační hodnota takového pole je pak nulová.

2.4.5 LSP

Formát LSP hlavičky je vyobrazen na obrázku 2.5. LSP hlavička obsahuje (mimo obecnou



Obrázek 2.5: Formát LSP hlavičky

hlavičku):

Délka PDU

Určuje délku v bytech celého PDU včetně hlavičky.

Životnost

Doba v sekundách, po kterou je daný LSP platný v rozsahu 2^{16}.

LSP-ID

LSP-ID je složené z položky *System-ID* popsané v části 2.3.3 a dvou bytů reprezentující *Pseudonode-ID* a *Fragment-ID*. Obě položky si podrobně popíšeme později. Pro jednoduchost si uveďme alespoň, že *Pseudonode-ID* se používá při komunikaci na LAN pro označení všech IS na segmentu. *Fragment-ID* se používá při fragmentaci LSP větších než MTU.

Sekvenční číslo

32-bitová hodnota určující sekvenční číslo podrobněji zpracováno v části 2.7.

Kontrolní součet

Kontrolní součet obsahu LSP od položky *LSP-ID* do konce. Počítá se podle algoritmu vytvořeném Johnem G. Fletcherem specifikovaným v rámci ISO/IEC 8473-1 [11] a ITU-T X.233 [15].

P/ATT/LSPDBOL/IS Type

Jedná se o blok (1 B) sdružených atributů.

První bit P označuje, zda daný IS podporuje funkci *Partition Repair*. Jedná se o volitelnou funkci protokolu IS-IS, která nemusí být implementována. Pokud selháním některé linky dojde k rozdělení *Level 1* oblasti, může se opravit dočasným využitím spojení mezi *Level 2* IS. Jedná se o poměrně složitý mechanismus a protože není povinnou součástí, implementace jej většinou ignorují.

Pokud jsou následující 4 b nastaveny, vyjadřují, že daný IS je připojen do okolních sítí, a používá některou z následujících metrik:

- 7. bit Error Metric,
- 6. bit Expense Metric,
- 5. bit Delay Metric,
- 4. bit Default Metric.

LSPDBOL se nastavuje pokud dojde k vyčerpání paměti daného IS. Ostatní IS při výpočtu nejkratší cesty vyřadí takové cesty, které by používaly daný IS jako tranzitní. Výhodou je, že přímo připojené sítě daného IS zůstávají stále dostupné. Poslední dva bity - IS Type - určují topologii, které je daný IS součástí. Každý router je součástí L1 topologie. Pokud je nastaven i druhý bit, je součástí i L2 topologie. Hodnota *IS-Type* pro takový IS by byla `3`.

2.4.6 Sequnce Numbers PDU

Sequnce Numbers PDU, respektive *Complete Sequence Number PDU* a *Partial Sequence Number PDU* slouží pro synchronizaci LSP databáze. CSNP zprávy na Ethernetu jsou zasílány na adresy `0180:c200:0014` (všechny L1 IS) a `0180:c200:0015` (všechny L2 IS) podle odpovídající úrovně. Na point-to-point linkách se CSNP zprávy používají pouze na prvotní synchronizaci při vytvoření sousedství. Následnou synchronizaci zajišťují PSNP zprávy. Oproti broadcastovým rozhraním, kde se PSNP používá pro zaslání chybějícího LSP, na point-to-point lince se využívá jako potvrzení. Formát CSNP hlavičky je na obrázku 2.6. CSNP hlavička obsahuje:

PDU Length

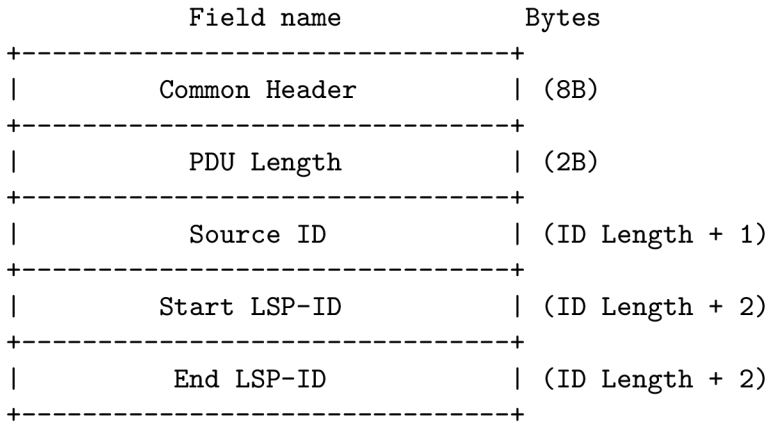
PDU Length udává délku celé CSNP zprávy včetně obecné hlavičky, CSNP hlavičky a samotných dat v TLV části.

Source-ID

Délka *Source-ID* se odvíjí od délky *System-ID* uvedené v obecné hlavičce, rozšířené o nulový byte reprezentující *Circuit-ID*.

Start LSP-ID a End LSP-ID

Stejně jako velikost *Source-ID*, tak velikost *Start LSP-ID* i *End LSP-ID* je závislá na délce *System-ID* z obecné hlavičky. V tomto případě rozšířené o 2 B – *Circuit-ID* a *Fragment-ID*. Pokud se podaří vměstnat všechny záznamy z link-state databáze (posílají se pouze hlavičky



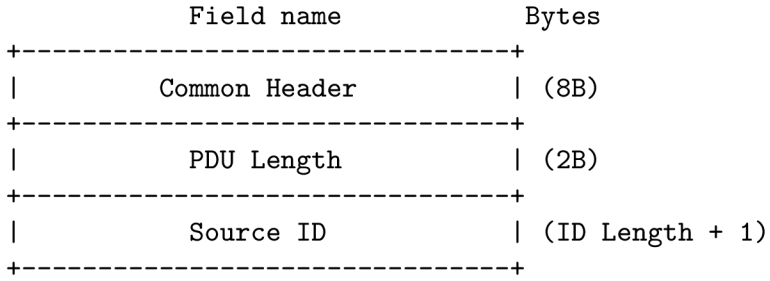
Obrázek 2.6: Formát CSNP hlavičky

LSP), nastaví se *Start LSP-ID* na hodnotu `0000.0000.0000.00-00` (binárně samé nuly) a *End LSP-ID* na `FFFF.FFFF.FFFF.FF-FF` (binárně samé jedničky). Tímto způsobem dá odesílatel příjemci najevo, že daný CSNP obsahuje celý rozsah LSP z odesílatelovy databáze. V případě, že se všechny záznamy nevejdou do jednoho CSNP, nastavíme položku *Start LSP-ID* prvního CSNP na `0000.0000.0000.00-00` a *End LSP-ID* na hodnotu posledního LSP z TLV části. U dalších CSNP nastavíme *Start LSP-ID* na hodnotu prvního LSP z TLV části a *End LSP-ID* na hodnotu posledního LSP. U posledního CSNP nastavíme *End LSP-ID* opět na hodnotu `FFFF.FFFF.FFFF.FF-FF`, čímž příjemce identifikuje poslední CSNP. Příjemce také předpokládá, že odesílané LSP záznamy jsou seřazeny podle *LSP-ID*.

V TLV části CSNP zpráv se mohou objevit pouze dva typy – #9 (LSP Entries) a #10 (Authentication). Autentizaci si popíšeme později s dalšími rozšířeními IS-IS. V CSNP neposíláme celé LSP, ale pouze hlavičku původního LSP. Struktura TLV #9 *LSP Entry* je vyobrazena na obrázku C.5. Význam jednotlivých polí je vysvětlen v sekci 2.4.5.

Pomocí těchto čtyř parametrů můžeme jednoznačně identifikovat LSP v link-state databázi.

PSNP hlavička je skromnější a kromě délky, obsahuje pouze *System-ID* zdrojového IS viz obrázek 2.7.



Obrázek 2.7: Formát PSNP hlavičky

ISO 10589:2002 definuje pro všechny LSP dva příznaky na každou linku pro lepší kontrolu zasílání LSP aktualizací. Jedná se o SRM (Send Routine Message) a SSN (Send Sequence Numbers). Oba příznaky jsou čistě lokální informací a neposílají se v žádné zprávě ostatním IS. Pokud nastavíme SRM příznak, znamená to, že odpovídající LSP musí být za-

sláno na danou linku. Pokud je nastaven SSN příznak, měl by být odpovídající LSP zahrnut v další zasílané PSNP zprávě.

2.5 Navazování sousedství

K úspěšnému navázání sousedství je potřeba zjistit, jestli je daná linka obousměrná. IS se musí ujistit, že zprávy na dané lince mohou proudit oběma směry před tím, než začne propagovat danou linku ve svých LSP. IS-IS používá dva způsoby ověření linky:

- 2-way handshake a
- 3-way handshake.

Jak už název napovídá, hlavní rozdíl je v počtu zpráv, které se musí úspěšně vyměnit.

2.5.1 2-way handshake

Pro úspěšné ověření obousměrného fungování linky tak stačí IIH zpráva od obou IS a sousedství na dané lince je prohlášeno za úspěšné. Problém je, že nevíme, zda nám IIH zpráva přišla jako odpověď na námi odeslanou Hello zprávu, nebo ji druhá strana vyslala na základě své vlastní činnosti.

2.5.2 3-way handshake

U *3-way handshake* je tomu jinak. Při příjmu Hello zprávy odpovídáme Hello zprávou v které je pomocí speciální TLV položky uvedeno na jakou Hello zprávu odpovídáme. Ten při příjmu této zprávy prohlásí sousedství za úspěšně navázané. My však musíme počkat na zprávu, kde bude v TLV uvedeno, že se jedná o odpověď na naší zprávu. Tím, že uvádíme, na jakou IIH zprávu odpovídáme zajišťujeme *stavovost*. Tento způsob je daleko robustnější, nicméně vyžaduje jednu zprávu navíc.

To, který způsob ověření obousměrnosti linky IS-IS použije, záleží na typu linky.

Ještě předtím než IS-IS vyzkouší, zda je linka obousměrná, otestuje linku, aby zjistil, zda podporuje velké pakety pro pozdější výměnu dat.

2.5.3 Ověření MTU

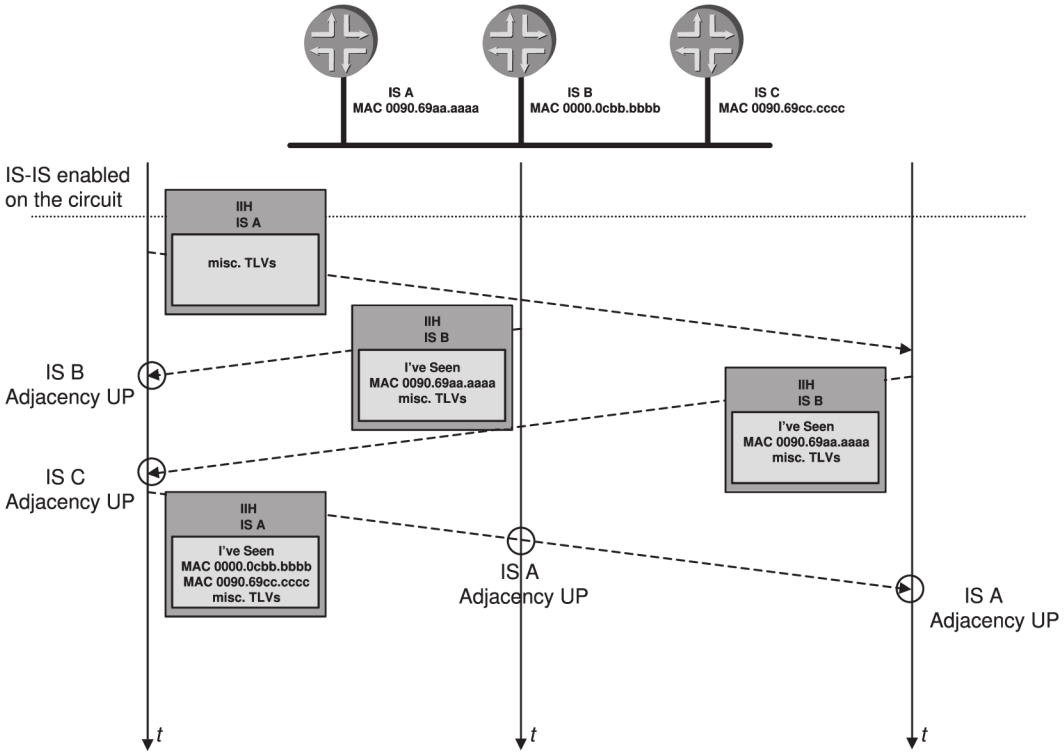
Největší paket, který by IS-IS mohl přenášet je 1492 B velký. IS-IS používá k ověření velikosti MTU uměle nafouknutou IIH zprávu pomocí TLV číslo 8. ISO 10589:2002 nedefinuje, zda po úvodním ověření linky, můžeme používat IIH s běžnou velikostí. Proto se některé implementace liší. IOS uměle zvětšuje všechny IIH zprávy, na rozdíl třeba od JUNOS, který posílá *nafouknuté* IIH pouze dokud se neustálí sousedství, poté posílá již normální IIH zprávy, čímž zbytečné nezatěžuje přenosové pásmo.

2.5.4 3-way handshake na LAN sítích

Na LAN sítích IS-IS používá 3-way handshake. Jelikož IIH zpráva dojde všem IS na LAN segmentu, nestačí pouhá odpověď od všech IS, které tuto zprávu dostali. V odpovědi na takovou IIH zprávu používáme TLV #6. Pro ukázku, mějme tři IS IS-A, IS-B a IS-C viz obrázek 2.8. Zmíněný obrázek byl s drobnými úpravami převzat z [9]. IS-A zašle IIH zprávu, kterou obdrdží IS-B i IS-C. Oba odpoví IIH zprávou s TLV #6, kde uvedou SNPA

(SubNetwork Point of Attachment) IS-A. SNPA pro LAN sítě je MAC adresa. IS-A po přijetí těchto zpráv ví, že IS-B i IS-C dostaly jeho IIH zprávu a nastaví si sousedství s těmito IS za navázané. IS-A ještě informuje IS-B a IS-C, že přijal jejich zprávu pomocí IIH, kde uvede MAC adresy obou IS. Po přijetí zprávy oběma IS je sousedství prohlášeno za navázané i oběma IS-B i IS-C.

Zmiňované TLV #6 se nazývá *IS Neighbor* TLV #6. Obsahuje seznam šesti-bytových SNPA, tedy MAC adres. Délka tohoto TLV proto musí být vždy násobek 6, jinak se jedná o poškozený paket.



Obrázek 2.8: Průběh navazování spojení pomocí 3-way handshake na LAN.

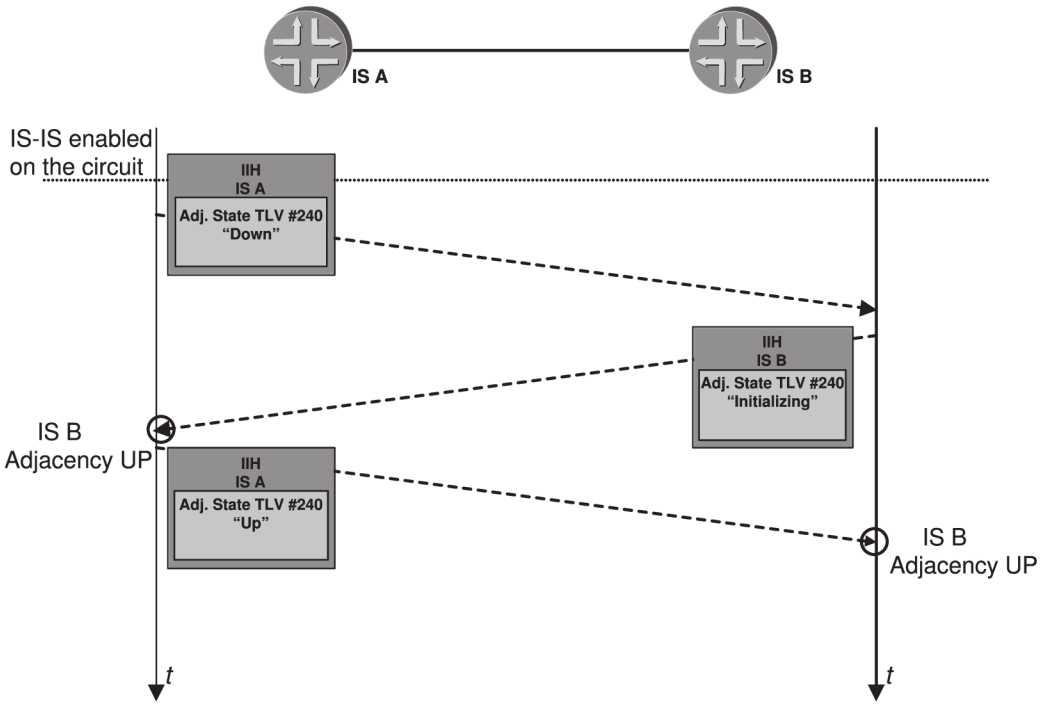
Na LAN sítích máme k dispozici pouze 3-way handshake. Na point-to-point linkách máme možnost výběru mezi 2-way a 3-way handshakem, nicméně používání 2-way handshaku je silně nedoporučeno.

2.5.5 3-way handshake na point-to-point lince

Na rozdíl od 3-way handshake na LAN sítích nemůžeme použít TLV #6, protože je navržené na míru právě LAN prostředím. Informace, kterou potvrzuje přijatou Hello zprávu MAC adresa. Nicméně point-to-point protokoly jako PPP, HDLC, Frame-Relay, nebo ATM nepoužívají MAC adresy. Na point-to-point linkách většinou nepotřebujeme žádné adresování, protože na dané lince jsou pouze dvě komunikující entity. Překlenutí tohoto problému je popsáno v RFC 5303 [16], kde je navrženo speciální TLV #240 *Point-to-Point Three-Way*

Adjacency State. Hlavním úkolem tohoto TLV je zjištění, zda IIH, kterou daný IS přijal, je odpověď na předchozí IIH, nebo obecné IIH zaslané druhým IS. Pokud zjistíme, že je to odpověď na naši předchozí IIH zprávu, můžeme předpokládat, že daná linka je obousměrná.

Průběh navazování sousedství pomocí TLV #240 můžeme vidět na obrázku 2.9. Zmíněný obrázek byl s drobnými úpravami převzat z [9]. Původní specifikace IS-IS požaduje



Obrázek 2.9: Průběh navazování spojení pomocí 3-way handshake na point-to-point.

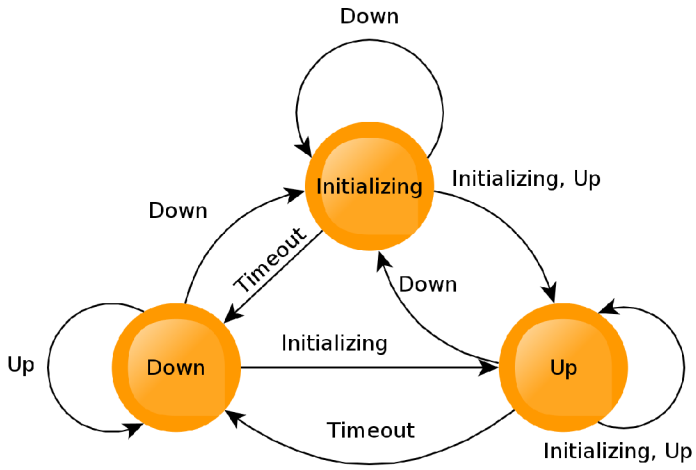
spolehlivý protokol linkové vrstvy na *point-to-point* linkách. Aby se mohl IS-IS používat i na linkách, který tento požadavek nesplňují, bylo zavedeno rozšíření v podobě *Point-to-Point Three-Way Adjacency* TLV #240. Jeho struktura je uvedena na obrázku 2.10. Záměrem je poskytnout *three-way handshake*, který bude zpětně kompatibilní. TLV #240 může být dlouhé 1, 5, 11, nebo 15 B. Minimálně 1 B

- **Adjacency Three-Way State** má jednu z následujících hodnot
  - 0 - Up
  - 1 - Initializing
  - 2 - Down
- **Extended Local Circuit ID** je jedinečný identifikátor přidělený dané lince odesílajících IS.
- **Neighbor System ID** je *System-ID* odesílajícího IS.
- **Neighbor Extended local Circuit ID** je *Extended Local Circuit ID* přidělené lince druhým IS.

```
            Field name                    Bytes
+-----------------------------------+
| Adjacency Three-Way State         | (1B)
+-----------------------------------+
| Extended Local Circuit ID         | (4B)
+-----------------------------------+
| Neighbor System ID                | (ID Length B)
+-----------------------------------+
| Neighbor Extended local Circuit ID| (4B)
+-----------------------------------+
```

Obrázek 2.10: Formát TLV #240 Point-to-Point Three-Way Adjacency

Všechny IS, které podporují toto rozšíření by jej měly používat ve svých PTP Hello. Na obrázku 2.11 je znázorněn přechodový automat navazování sousedství při použití TLV #240 na *point-to-point* linkách.



Obrázek 2.11: Stavový automat navazování sousedství s použitím TLV #240.

2.6 Link-state databáze

Link-state databáze reprezentuje kompletní topologii dané oblasti. Každý IS v dané oblasti postupně rozesílá své a přeposílá LSP od ostatních IS. Po ustálení tak všechny IS v dané oblasti mají stejnou databázi. K vytvoření kompletní mapy oblasti IS-IS používá několik metod zvaných zaplavování (flooding) a synchronizace (synchronizing). IS-IS si vytváří dvě databáze pro ukládání topologických informací. První pro reprezentaci IS v nejbližším okolí, nazývanou point-of-presence (POP). Jedná se o databázi L1 IS. V druhé databázi je uložena celková topologie reprezentující IS na L2. Díky tomu, že každý IS má kompletní/stejnou

databázi, může vypočítat topologii sítě nezávisle na ostatních IS. Tomuto principu říkáme *místní výpočet*.

2.6.1 Místní výpočet

V případě distance-vector směrovacího protokolu výpočet nejlepší cesty probíhá distribuovaným způsobem. Pokaždé, když RIP router přeposílá informace o dostupné síti, zhorší její dostupnost (metriku) o počet hopů, většinou tedy zvýší hodnotu o 1. Nejenom díky tomu, že zvýšení metriky nemusí být pouze o 1, ostatní routery v sítí netuší s jakou metrikou jsou dané sítě dostupné pro určitý router. V link-state protokolu informace o dostupných sítích pouze přeposíláme, ale neupravujeme. Po úvodním rozeslání zpráv (zaplavení) všem IS v oblasti, IS-IS spustí výpočet cest pomocí algoritmu shortest path first zkráceně SPF. Každý IS provádí tento výpočet samostatně. Způsobu výpočtu nejkratších cest je věnována část 2.18.1.

2.7 Číslování LSP

Abychom zjistili, v jakém pořadí byly jednotlivé zprávy generovány, a tím určili, která je poslední a tedy nejaktuálnější, používáme sekvenční čísla.

Potřebujeme, aby LSP obsahoval informaci, která by vyjadřovala co je aktuální a co zastaralé.

Podobně jako je tomu u TCP i v IS-IS se využívá sekvenčních čísel. Pomocí nich dokážeme vyjádřit o jakou verzi link-state databáze se jedná. IS-IS používá 4 B pole *Sequence number* s výchozí hodnotou `1`. První odeslaný LSP bude mít tedy hodnotu sekvenčního čísla `0x1`.

Pokaždé, když IS oznamuje změnu v topologii svým sousedům, inkrementuje hodnotu sekvenčního čísla v LSP o jedničku. IS, který danou zprávu přijal, zkontroluje, zda od tohoto zdroje nějaký LSP již obdržel, pokud ne, nainstaluje daný LSP do místní link-state databáze. V případě, že už od daného zdroje nějaké LSP má, tak musí zkontrolovat, jestli přijaté sekvenční číslo je vyšší než aktuální hodnota. Pokud je přijatá hodnota vyšší, nahradí, případně vymaže, existující LSP nově přijatým. Pokud je hodnota nižší, přijatý LSP se jednoduše zahodí. Jelikož je IS-IS *spolehlivý* protokol, potvrzují se i LSP, které zahazujeme.

Maximální hodnota sekvenčního čísla je tedy 2^{32}. V případě zasílání LSP každých 5 s by nám to stále stačilo na zhruba 681 let. Rozsah je tedy poměrně velký a dalo by se předpokládat, že nebude nikdy vyčerpán. Ovšem pro případ, že by k takové situaci přece jen došlo, obsahuje každý LSP položku *Lifetime*, životnost.

2.8 Životnost LSP

Všechny LSP pro určení platnosti obsahují mimo sekvenčního čísla ještě pole životnost. *Životnost* zajišťuje platnost daného LSP pouze po určitou dobu. Pomáhá tak s odstraňováním zastaralých a potencionálně chybných LSP z link-state databáze. V LSP je pro životnost vyhrazeno 16 b udávajících hodnotu v sekundách s maximální dobou 65535 s.

Využíváním životnosti LSP dochází ke stárnutí již nainstalovaných LSP v link-state databázi. Je proto nutné položky pravidelně obnovovat pomocí periodických aktualizací.

2.9 Periodické aktualizace

Periodická aktualizace znamená, že IS musí pravidelně odesílat již jednou odeslané LSP. Interval mezi znovuodesláním LSP musí být samozřejmě menší, než je doba životnosti daného LSP. Při každé aktualizaci se inkrementuje hodnota sekvenčního čísla. Doporučený interval periodické aktualizace dle ISO 10589:2002 je 1200 s.

Životnost i interval periodické aktualizace můžeme nastavovat nezávisle na sobě, ale musíme zajistit, aby interval periodické aktualizace byl vždy menší než životnost. Nejlépe o desítky sekund, abychom vykompenzovali případné zpoždění, nebo výpadek.

2.10 Expirace LSP

Pokud není LSP včas obnoven periodickou aktualizací, nebo není vynuceno jeho odstranění pomocí přijetí prázdného LSP, dojde k vypršení doby *životnosti* - expiraci daného LSP. Pokud se hodnota některého LSP dostane na `0` dojde k jeho odstranění z link-state databáze. Aby byla zajištěna konzistence mezi link-state databázemi všech IS, iniciuje IS odeslání LSP, který způsobí odstranění i z ostatních IS viz sekce 2.13. Ačkoliv by měla životnost daného LSP vypršet ve stejnou dobu na všech IS, je kvůli možnosti rozsynchronizování hodin tato vlastnost vynucena explicitně. V normálním případě by k vypršení LSP nemělo nikdy dojít, protože buď jej IS obnoví pomocí pravidelné aktualizace, nebo si původce daného LSP vynutí jeho odstranění. Ovšem, ani po vyžádání odstranění, ani po expiraci životnosti není LSP okamžitě odstraněn. Po vypršení doby životnosti LSP je daný LSP udržován v link-state databázi po dobu značenou jako *Zero Age Lifetime*. Po tuto dobu se LSP nachází v databázi, ale není zahrnováno do výpočtu nejkratších cest. Výchozí *Zero Age Lifetime* je 60 s.

2.11 Zaplavování

Zaplavování (z anglického výrazu *flooding*) je mechanismus rozesílání LSP svým sousedům. Pracuje ve dvou verzích podle toho, zda je původcem LSP, nebo jej pouze přeposílá. V případě, že je původcem LSP, odešle jej na všechna rozhraní, na kterých má úspěšně navázaná sousedství. Pokud LSP pouze přeposílá, zkontroluje nejdříve sekvenční číslo pro ověření aktuálnosti. Pokud je hodnota vyšší než aktuálně nainstalovaná v link-state databázi, nainstaluje nový LSP a přepošle jej na všechna rozhraní kromě toho, na kterém daný LSP přijal. Díky kontrole sekvenčního čísla při přeposílání zabráníme jevu *LSP bouře* (*LSP storm*), kdy dochází k nekontrolovanému zvyšování přeposílaných zpráv, až dojde k zahlcení sítě.

Ačkoliv kontrolováním sekvenčního čísla při předávání LSP zabráníme nekonečnému putování LSP po síti, na mesh sítích dochází ke zbytečnému přeposílání. Pokud bychom měli full-meshed síť, tak již při *záplavě* LSP od zdroje dojde k informování všech IS. Ti ale nemají tušení o tom, jakou zprávu dostali ostatní sousedé, a proto přepošlou daný LSP všem sousedům, kromě původního zdroje. První IS tedy rozeslal $N - 1$ zpráv, na které $N - 1$ IS odpovědělo $N - 2$ zprávami. Při fully-meshed síti se dostáváme až k prostorové složitosti $O(N^2)$ při rozesílání aktualizací.

Jakým způsobem můžeme zabránit několikanásobnému přeposílání LSP na jeden IS z více sousedů?

2.12 Mesh Groups

Odpovědí na tento problém je RFC 2973 IS-IS Mesh Groups. Jde o způsob prořezávání topologie jako je tomu např. u Spanning Tree. To, které části topologie mají být prořezány, musíme určit ručně. Je nutné si dávat pozor, abychom topologii neprořezali příliš těsně, protože se neumí sama adaptovat při výpadku některé linky. Ačkoliv Mesh Groups je zajímavý způsob, jak snížit množství LSP, je to spíše záležitost minulosti při využívání ATM a Frame Relay sítí.

V dnešní době, především kvůli statické konfiguraci a neschopnosti automatické opravy, se příliš nevyužívá. Místo Mesh Groups se na broadcastových sítích využívá *Pseudonode*.

2.13 Odstranění LSP

Pro odstranění LSP bychom se mohli spolehnout na vypršení *životnosti* daného LSP, ale to by při maximální hodnotě mohlo trvat až 18 h. V případě, že chceme bezpečně odebrat IS z dané topologie, tedy ze všech link-state databází, iniciujeme *Network Wide Purge*.

Jedná se o LSP, které má vynulované položky *Remaining lifetime* a kontrolní součet. Sekvenční číslo takového LSP je stejné, nebo větší než poslední odeslané LSP. IS při přijetí takového LSP vymaže všechny LSP od daného IS ze své link-state databáze.

Network Wide Purge se využívá při volbě nového ***D**esignated **I**ntermediate **S**ystem* (DIS).

2.14 Volba DIS

Na broadcastových LAN sítích má jeden IS speciální funkci. Slouží jako *Designated Intermediate System* (DIS). DIS používá *LAN-ID*, které je jedinečné v rámci dané LAN, a šíří jej všem ostatním IS na segmentu. *LAN-ID* je *System-ID* daného IS rozšířené o 1 B. V případě volby nového DIS potřebujeme odstranit LSP záznamy od původního LSP. Původní DIS tedy vygeneruje a rozešle LSP s nulovými položkami pro životnost a kontrolní součet a hodnotu sekvenčního čísla zvýší o jedničku. Každý IS, který tento LSP přijme, odstraní uvedené *LSP-ID* ze svojí link-state databáze.

2.15 Pseudonode a DIS

Na broadcastových sítích, kde každý *vidí* každého, si IS-IS musí vytvářet velké množství záznamů v link-state databázi a dochází k výměně velkého počtu Hello PDU. Aby se zmenšil dopad těchto problémů zavádí protokol IS-IS pojmy *Pseudonode* a *Designated Intermediate System* (DIS). V následující sekci si popíšeme, v kterých situacích je tento koncept vhodný, jakým způsobem probíhá volba DIS, preempce a další.

2.15.1 Vyvážení počtu sousedství na rozsáhlých LAN

Kdykoli máme velký počet IS na LAN je potřeba vzít v úvahu množství *mluvčích* na segmentu a s tím související počet Hello zpráv. Pokud do stávající rozsáhlé sítě přidáme dalšího mluvčího (IS), dojde ke generování velkého množství zpráv - nový mluvčí (po 3-way handshaku [16]) zašle všem Hello zprávu, ostatní mu na ní odpoví, a vygenerují LSP se změnou

v síti. Pokud by všechny IS odpověděly okamžitě, docházelo by nárazově k obrovskému nárůstu zpráv. A v případě, že všichni používají stejný (např. výchozí) *Hold time*, docházelo by k těmto *bouřím* pravidelně. Tomuto jevu se říká *self-synchronization problem.*

2.15.2 Problém synchronizace

Abychom zabránili generování odpovědí, Hello zpráv a periodických aktualizací ve stejný čas a přitom zajistili dodržení časových limitů specifikuje ISO 10589:2002 povinně *jitter* (odchylku) 25 %. Tato odychlka se generuje náhodně a měla by odpovídat uniformnímu rozložení během celé doby generování. Vygenerovaná hodnota se odečítá od původní hodnoty daného časovače.

2.15.3 Pseudonode

Ve chvíli, kdy máme skupinu IS připojených na jednom segmentu LAN, dochází k vytváření sousedství každý s každým, což má prostorovou složitost $O(N^2)$. Roste tedy exponenciálně s počtem IS. Způsob jakým tento problém vyřešili tvůrci IS-IS spočívá v pojmu *Pseudonode.* Jedná se o reprezentaci dané LAN jako dalšího uzlu. Protože *pseudonode* jsou pouze dráty propojující IS bez jakékoliv logiky nutné pro provádění nezbytných úkonů v IS-IS, musí jej zastoupit některý skutečný IS. Takový IS označujeme jako *Designated Intermediate System* (DIS). DIS je jedním z IS na dané LAN a je označen speciálním *System-ID*, díky kterému poznáme, že se jedná právě o *Pseudonode.* Strukturu a způsob vytváření *System-ID* pro DIS si popíšeme později. Použitím *Pseudonode* výrazně redukujeme původní složitost každý s každým $O(N^2)$ na hvězdicovou topologii s $O(N)$. Nyní si popíšeme formát a způsob generování *System-ID* pro *Pseudonode.*

2.15.4 Pseudonode-ID

Každý IS má nakonfigurované *System-ID* např. 6 B. V LSP se ale posílá *LSP-ID* s dvěma byty navíc oproti *System-ID.* Poslední byte řeší problém fragmentace. Předposlední byte označujeme jako *Pseudonode-ID.* Prvních 7 bytů v *LSP-ID* bývá označováno jako *Node-ID* - identita uzlu.

Pokud je hodnota *Pseudonode-ID* nulová, znamená to, že se jedná o skutečný IS, naopak nenulová pak indikuje, že se jedná o *pseudonode. Node-ID* je tvořeno *System-ID* DIS pro danou LAN a jedním bytem (*Pseudonode-ID*), který zaručí unikátnost mezi *pseudonode* a skutečným IS, který jedná jeho jménem. Díky osmi bitovému poli *Pseudonode-ID* může daný IS jednat *ve jménu* až 255 pseudonodů. V případě, že se daný IS nechce účastnit volby DIS, nastaví *Pseudonode-ID* na `0`.

2.15.5 Modelování link-state databáze

Každé sousedství na LAN je ohodnoceno určitou cenou. Ve chvíli, kdy *DIS* vytvoří pseudonode, musí zajistit, že celková cena skrz LAN nebude zkreslena. Řešením je použití asymetrické ceny pro cestu k a *od* pseudonode. Původní cena je nastavena ve směru od reálných IS do pseudonode a pro cestu z pseudonode do IS je nastavena nulová cena. Pro reálné IS je nulová cena neplatná hodnota. Při výpočtu SPF proto musíme pseudonodu věnovat speciální pozornost.

Pseudonody byly navrženy pro snížení zátěže na IS, ale v některých situacích generování a údržba pseudonodu naopak přidává další zatížení.

V RFC 5309 [24] je popsán způsob, jak se vyhnout generování pseudonodu. Návrh spočívá v poslání point-to-point Hello zpráv i na broadcastovém spojení. Pokud se oba (všechny) IS shodnou, nedochází k volbě DIS ani generování pseudonodu.

Ještě předtím, než dojde ke generování pseudonode, musí být na LAN přítomen DIS.

2.15.6 Volba DIS

Volba DIS je bezstavová a jsou pro ni vyhrazeny dvě pole v LAN IIH hlavičce – *Priority* (priorita) a *Source SNPA* (MAC adresa na Ethernetu). Priorita může být v rozsahu `1 - 127`. Hodnota `0` znamená, že daný IS nechce být *DIS*. Pokud je na segmentu více IS se stejnou *Priority*, rozhoduje vyšší *Source SNPA*. Jednotlivé IS provádí výpočet lokálně pří přijetí Hello zprávy porovnáním s aktuální *DIS* prioritou. K volbě dochází *pre-emptivně*. Pokud se připojí IS s vyšší prioritou, původní DIS rezignuje. Pro zdokumentování své akce uvede *Node-ID* nového *DIS* v poli *LAN-ID*. Následně musí odstranit původní pseudonode z link-state databáze. Zašle tedy *Purge LSP*, který obsahuje pouze hlavičku a nulové hodnoty pro *Lifetime* a *Checksum*.

Na rozdíl od protokolu OSPF, v IS-IS není záložní *DIS*. Pokud dojde k výpadku spojení s *DIS*, musí proběhnout nová volba. Oproti OSPF má IS-IS výhodu, že může nastavit *Hold time* pro *DIS* na menší hodnotu, čímž zajistí, že výpadek *DIS*, bude detekován v kratším čase oproti ostatním IS. U OSPF bychom si to dovolit nemohli, protože *Hold time* musí být stejný pro všechny routery na LAN, což by výrazně zvýšilo zátěž sítě.

2.16 Synchronizace databází

Link-state protokoly jsou závislé na faktu, že všechny IS (směrovače) v dané oblasti mají přehled o stejné topologii. Pokud by neměly přehled o stejné topologii mohlo by při směrování docházet ke zbytečně dlouhým cestám, případně vytvoření smyčky. V měnící se sítí je potřeba synchronizovat topologii co nejrychleji. Jakým způsobem je toho docíleno v IS-IS si popíšeme v následující sekci.

Synchronizované link-state databáze a výsledné směrovací tabulky jsou zásadní pro směrování paketů do jejich cíle. Pro zajištění synchronizace link-state databází používá IS-IS dva speciální typy zpráv – CSNP (Complete Sequence Number Packet) a PSNP (Partial Sequence Number Packet). Způsob jejich použití je závislý od typu linky, zda se jedná o point-to-point, nebo broadcastová LAN.

DIS získá ze své link-state databáze všechny záznamy a naplní jimi odpovídající počet CSNP zpráv. Následné každý IS na LAN porovná záznamy ze své databáze s těmi přijatými od DIS. Pokud se u přijatých záznamů shoduje sekvenční číslo, všechno je v pořádku. Pokud ne, přijatý LSP záznam může být *starší*, *novější*, nebo *neznámý*.

V případě příjetí staršího LSP je řešení jednoduché. Protože se zdá, že DIS nemá aktuální informace, pošleme poslední verzi daného LSP znovu na LAN.

Pokud je přijatý LSP záznam novější, musí příjemce nastavit SRM příznak pro tento LSP, což způsobí zaslání PSNP zprávy pro DIS. PSNP zprávy mají *PDU Type* 24 pro L1 a 25 pro L2. Požadované LSP jsou do PSNP přidány pomocí TLV #9. DIS po přijetí takové PSNP zprávy jednoduše zašle nejnovější verzi požadovaných LSP na LAN.

Nakonec v případě, že DIS zasílá nový, nebo neznámý LSP záznam, příjemce opět odpovídá PSNP zprávou. Jelikož nemá žádné informace o daném LSP, signalizuje tuto situaci vynulováním položek *Sequence number*, *Lifetime* a *Checksum*. Reakce na takovou

PSNP zprávu je stejná jako v případě žádosti o zaslání pouze novější verze. DIS jednoduše zašle nejnovější verzi požadovaných LSP.

Zmiňované PSNP má jednodušší formát než CSNP. V hlavičce PSNP zprávy, kromě obecné hlavičky, je pouze *PDU Length* a *Source-ID*. V TLV části používá stejně jako CSNP TLV #9.

S využitím *DIS* je synchronizace databází na LAN velice jednoduchá a čistá. Kromě dvou příznaků, SRM a SSN nepotřebuje v podstatě žádné jiné stavové informace.

2.16.1 Synchronizace databází na point-to-point

Pro synchronizaci databází na point-to-point linkách používáme také CSNP a PSNP, ale mají zde jiný význam než na broadcastových sítích.

V okamžiku navázání sousedství mezi dvěma IS si oba vzájemně zašlou obsah svojí databáze prostřednictvím CSNP. Dojde tak k prvotní synchronizaci databází obou IS. Pokud příjemce (IS-B) zjistí, že odesílatel (IS-A) má novější verzi určitého LSP, nebo dané LSP vůbec nezná, negeneruje žádnou akci. Ve chvíli, kdy se situace obrátí, a IS-B bude odesílatelem CSNP zprávy, IS-A detekuje starší verzi, případně absenci daného LSP, a sám zašle dané LSP. PSNP se pak použije jako potvrzení přijetí daného LSP a zajištění tak *spolehlivé* komunikace. Při odeslání určitého LSP, si odesílatel nastaví SRM příznak, jež signalizuje nutnost zaslání daného LSP, a nezruší ho, dokud mu nepřijde potvrzení ve formě PSNP. IS v pravidelných intervalech kontroluje SRM příznak a rozesílá dokud je LSP nepotvrzené.

2.17 Fragmentace

Jak již bylo několikrát zmíněno, IS-IS se od ostatních směrovacích protokolů liší v mnoha ohledech. Na rozdíl od protokolu OSPF, který běží na vrstvě IP modelu TCP/IP stacku, případně BGP, který využívá dokonce služeb TCP, se protokol IS-IS nemůže spolehnout na služby nižších vrstev zajišťujících posílání zpráv větších než MTU dané linky. Protože běží přímo na druhé vrstvě modelu OSI, musí být možnost zasílání zpráv delších než MTU dané linky zahrnuta přímo do protokolu IS-IS.

IS-IS využívá dva ze tří způsobů používaných v *IP – fragmentace na síťové vrstvě* a *předpokládané minimální MTU*.

Předpokládané minimální MTU vychází z faktu, že ISO 10589:2002 stanovuje minimální MTU, které musí linka splňovat, aby se na ní mohl provozovat IS-IS. V případě, že daná linka nepodporuje MTU minimálně 1492 B, nedojde k navázání sousedství. Kontrola MTU probíhá v *handshake* fázi posíláním uměle nafouknutých IIH zpráv. Nevýhoda tohoto řešení je, že na linkách s větším MTU dochází k plýtvání přenosovým pásmem zvýšenou režií.

Další způsob řešení je použití fragmentace. Protože protokoly nižších vrstev nejsou schopny tuto činnost zajistit, musí ji implementovat přímo IS-IS.

IS-IS používá tři základní typy zpráv – Hello (IIH) pro objevování sousedů a výše zmíněnou kontrolu MTU, Sequence number packet (SNP) pro synchronizaci link-state databází a Link-state (LSP).

2.17.1 Hello zprávy

Co se týče IIH zpráv, IS-IS nepodporuje zpracování IIH zpráv rozprostřených přes několik paketů. V současnosti to neznamená žádný problém, protože průměrná velikost IIH zpráv

je v rozmezí od 40 – 70 B a všechny linky používané v IS-IS musí podporovat přenášení minimálně 1492 B. IIH obecně *trpí* spíše opačným problémem, kdy se při testování schopnosti linky IIH zprávy uměle zvětšují pomocí TLV části.

2.17.2 Sequence Number Packets

Sequence Number Packets zahrnují jak CSNP, tak i PSNP. PSNP jsou používány pro potvrzování přijatých LSP, nebo naopak pro vyžádání LSP uvedeného v TLV #9 LSP Entry. V obou případech může PSNP zpráva obsahovat více položek v LSP Entry. Je jedno, jestli odesílatel čeká určitou dobu a sdružuje více položek do jednoho záznamu, nebo je posílá jednotlivě. Ačkoliv záznamy v PSNP bývají seřazené, jejich pořadí nemá žádný vliv na jejich význam. Každý záznam v LSP Entry je zpracováván nezávisle na ostatních, proto není fragmentace PSNP potřeba

U CSNP zpráv nastává úplně jiná situace ve srovnání s PSNP. Při prvotní synchronizaci na point-to-point linkách, nebo pravidelných aktualizacích na LAN, zasíláme hlavičky všech záznamů z link-state databáze a příjemce musí vědět, které zprávy patří do jedné verze link-state databáze. Při špatné interpretaci by docházelo k opakovanému znovu posílání všech LSP záznamů, kromě těch přijatých v konkretním CSNP. Proto CSNP hlavička obsahuje *Start LSP-ID* a *End LSP-ID*. Pomocí těchto položek dokáže příjemce identifikovat začátek a konec jednoho setu CSNP zpráv. V menších prostředích, kde se obsah link-state databáze vleze do jedné CSNP zprávy se *Start LSP-ID* nastaví na samé nuly a *End LSP-ID* na samé jedničky (binárně). Příjemce tak pozná, že se jedná zároveň o první i poslední CSNP zprávu. Pro zamezení fragmentování již fragmentovaných dat, jsou hlavičky z link-state databáze rozděleny po maximálně 1492 B na jednu CSNP zprávu.

2.17.3 LSP

Pokud zmiňujeme potřebu fragmentace u zasílání jen hlaviček CSNP zpráv, musíme ji uvažovat i při zasílání celých LSP.

V LSP je fragmentace řešena pomocí 1 B pole *Fragment-ID*, které společně s *LAN-ID* (*System-ID* + *Pseudonode-ID*) tvoří dohromady *LSP-ID*. *Fragment-ID* udává, o kterou část původního LSP se jedná.

Pokud potřebujeme přenést LSP větší než MTU, rozdělíme jej vždy po celých TLV částech, a očíslujeme od 0. Následující části daného LSP mají položku *Fragment-ID* vždy o jedničku větší.

Přijaté fragmenty původního LSP jsou instalovány do link-state databáze příjemce a rozeslány sousedícím IS. IS-IS není závislý na přijetí všech fragmentů původního LSP v jedné iteraci distribuce LSP záznamů. Pokud některý ze záznamů nedorazí, dojde k jeho vyžádání pomocí synchronizačních mechanismů v podobě CSNP a PSNP zpráv. Předchozí tvrzení má jednu výjimku. Pokud nedorazí část s *Fragment-ID* `0`, ostatní fragmenty jsou zahozeny. Fragment 0 je důležitý, protože obsahuje některé informace, které mohou být pouze v nultém fragmentu, a jejich přítomnost určuje výskyt dalších parametrů ve všech následujících fragmentech. Bez fragmentu 0 nemůže být například zahájen výpočet SPF (Shortest Path First) algoritmu.

Stejně jako v dalších částech, tak i zde ISO 10589:2002 striktně nespecifikuje všechny nuance rozdělování LSP do jednotlivých fragmentů, ale nechává řešení na konkrétní implementaci. V nepromyšleném návrhu implementace může docházet ke zbytečnému generování a následnému fragmentování LSP. Jde o snahu negenerovat všechny fragmenty při výpadku

některého IS a následného posunu všech následujících TLV, čímž dojde ke generování i fragmentů, které by původním výpadkem nebyly ovlivněny. Ačkoliv i změna jediného fragmentu způsobí přepočítání SPF, šetříme prostředky IS nutné pro znovu generování LSP a následné rozesílání všem sousedícím IS.

Celkově můžeme díky 8 b poli *Fragment-ID* rozdělit původní LSP na 256 částí. Každý fragment může pojmout až 1470 B užitečných dat (bez LSP hlavičky), což nám dává celkem 376320 B. Při nasazení pro IPv4 může jediný router rozesílat kolem 42000 prefixů. Pro případ, že se jednoho dne narazí i na toto omezení, bylo zavedeno TLV #14 *LSP Buffer Size*, pomocí něhož si IS může požádat o zvýšení minimálního MTU z 1492 B.

2.18 Výpočet nejkratších cest

Už víme jakým způsobem navázat sousedství, ověřit linku, udržování spojeni/sousedství pomocí IIH, budováni link-state databáze pomocí LSP a její udržování pomocí CSNP a PSNP zpráv, ale jakým způsobem dostaneme z link-state databáze data relevantní pro směrování. Pro správné směrování a zajištění nejlepších cest, potřebujeme zajistit bezsmyčkovost do všech cílů v síti.

V této části si popíšeme Dijkstrův Shortest Path First algoritmus, určený pro grafy s nezápornými hranami, a jeho nasazení v rámci IS-IS. Zaměříme se na samotný výpočet SPF, route resolution a vkládání prefixů.

2.18.1 SPF

Shortest Path First, nebo také Dijkstrův algoritmus, je algoritmus vymyšlený Holandským vědcem Edsgarem Dijkstrou v roce 1956. Jedná se o algoritmus z oboru teorie grafů na výpočet nejkratších cest z daného uzlu do všech ostatních. Pro zajištění bezsmyčkovosti musí být ohodnocení všech hran v grafu nezáporné. Jednotlivé IS reprezentují uzly, linky mezi IS reprezentují hrany a metrika linky odpovídá ohodnocení hrany grafu.

SPF při výpočtu používá tři základní seznamy: UNKNOWN, TENT a PATH. Všechny uzly z link-state databáze jsou nejprve nakopírovány do seznamu UNKNOWN. Všechny dostupné IS aktuálně zpracovávaného uzlu se umístí do seznamu TENT počínaje záznamy uzlu z kterého počítáme cesty (kořen). Poté co SPF nalezne nejlepší cestu do daného uzlu, je tento uzel přemístěn do seznamu PATH. Seznam PATH je na začátku prázdný.

Základní iterace algoritmu spočívá v těchto krocích:

1. Najdi uzel s nejmenší cenou a přemísti jej do seznamu PATH.
2. Najdi všechny dosažitelné uzly z daného uzlu a přesuň uzly ze seznamu UNKNOWN do TENT.
3. Pro každý uzel, který je přesunut do seznamu TENT, udržuj cenu do daného uzlu a *first-hop*.

Při přesunu záznamu uzlu ze seznamu UKNOWN v kroku 2, musíme vždy provést nejprve kontroly obousměrnosti daného spoje. Pro záznam IS-A $->$ IS-B, se snažíme najít záznam IS-B $->$ IS-A. Pokud jej nenajdeme je takový záznam ignorován.

Zvláštní péči při výpočtu SPF musíme věnovat pseudonodu. Pokud máme v TENT seznamu více uzlů se stejnou cenou (nejmenší), upřednostníme při výběru pseudonode.

Zjednodušeně řečeno, do seznamu PATH přidáváme nejkratší cestu z TENT. Do seznamu TENT si dáváme kandidáty na nejkratší cestu, vždy když přidáme záznam do

TENT, přidáme z něho dostupné uzly (následníky) do seznamu TENT. Výpočet končí, pokud je seznam TENT prázdný.

Pro každou úroveň (L1 a L2) a každou metriku (default, expense, delay, error) je prováděn výpočet samostatně. Ačkoliv v návrhu IS-IS je počítáno s až čtyřmi metrikami podle nichž je možno směrovat, v praxi se tento model neuplatňuje a používá se pouze jedna. Pro L1L2 IS se všemi metrikami by bylo nutné provést výpočet SPF celkem osmkrát.

Nesmíme zapomenou na kontrolu overload bitu v nultém LSP daného IS. Pokud je tento příznak nastaven, dané cesty neuvažujeme, protože takový IS může mít nekonzistentní link-state databázi a mohlo by dojít k vytváření smyček při směrování.

Při spuštění výpočtu SPF musíme *zmrazit* aktuální stav link-state databáze.

Výpočet SPF probíhá ve dvou bězích. V prvním běhu se nejprve vytvoří topologická struktura oblasti čistě na základě informací z TLV *IS Reachability*. V druhém průchodu jsou zpracovávány ostatní informace z daného LSP.

2.19 Podpora na Cisco zařízeních

Protokol IS-IS podporuje většina směrovačů. Sada příkazů spojená s konfigurací IS-IS byla přidána do IOS verze 12.0. Způsob konfigurace si uvedeme v následující části, kde zmíníme nejdůležitější příkazy. Kromě popisu daného příkazu je uvedeno také ekvivalentní klíčové slovo pro konfiguraci IS-IS v rámci simulace v OMNeT++, případně popis odlišného chování.

2.19.1 Přehled konfiguračních příkazů

- `router isis [area-tag]` - global configuration - slouží k povolení IS-IS. `area-tag` je volitelný parametr označující daný proces. Pro vypnutí se před příkaz přidá `no`.
- `net net1 alt net2` - router configuration - nastaví NET identifikátor daného IS.
- `is-type [level-1 | level-1-2 | level-2-only]` -router configuration -nastaví na jaké úrovni má tento IS pracovat.
- `isis metric {metric-value | maximum} [level-1 | level-2]` - interface configuration - nastaví metriku pro dané rozhraní na hodnotu `metric-value`. Pomocí `level-1`, nebo `level-2` můžeme stanovit použití dané hodnoty pouze pro výpočet SPF vybrané úrovně.
- `clns router isis [area-tag]` - interface configuration -povolí IS-IS v režimu ISO CLNS na rozhraní. `area-tag` je volitelný parametr označující IS-IS proces.
- `isis circuit-type [level-1 | level-1-2 | level-2-only]` - interface configuration - nastaví úroveň pro sousedství, které mohou být na tomto rozhraní navázány.

Kompletní sada příkazů je uvedena v [1].

Kapitola 3

TRILL

V této kapitole se seznámíme s protokolem TRILL pracujícím na linkové vrstvě. Představíme nový typ zařízení, v rámci kterého je jeho činnost definována. Popíšeme podpůrné mechanismy, které jsou nutné pro správné fungování TRILLu. Nezapomeneme na vylepšení, která zavádí oproti stávajícím řešením a jakým způsobem se vyrovná s nasazením v nehomogenní topologii.

TRILL – *Transparent Interconnection of Lots of Links* je protokol linkové vrstvy zajišťující bezsmyčkovost. Jedná se o náhradu za zastarávající protokol STP – *Spanning Tree Protocol*. Stejně jako u vzniku STP, tak i u protokolu TRILL, stála v čele výzkumné skupiny Radia Perlman. Vývoj probíhal v rámci IETF a v červenci 2011 byl vydán jako IETF RFC standard. Jeho hlavní části jsou popsány v [22], [6] a [7]. Kromě zmíněných RFC jsme čerpali z [10].

Abychom mohli začít s popisem protokolu TRILL, musíme nejprve představit novou kategorii zařízení, v rámci kterého je funkce TRILLu definována. Zařízení, které máme na mysli, se označuje *RBridge – Routing Bridge*.

3.1 RBridge

RBridge poskytuje optimální přeposílání, a to i během dočasných smyček, podporu pro *multipathing* jak *unicast*, tak *multicast* provozu. Tohoto cíle je dosaženo použitím směrovacího protokolu IS-IS a zapouzdření daného provozu hlavičkou s položkou *hop count*.

RBridge je zpětně kompatibilní s přepínači dle IEEE 802.1 customer brigdes stejně jako s IPv4 a IPv6 směrovači a koncovými stanicemi. Pro protokoly vyšší vrstvy je RBridge topologie naprosto transparentní stejně jako je tomu u klasických přepínačů a stejně jako směrovače i RBridge tvoří hranice spanning tree protokolu.

V návrhu je zakomponována podpora VLAN a optimalizace distribuce *multi-destination* rámců podle VLAN ID a podle multicastových skupin založených na IP. Zároveň umožňuje, aby velikost forwardovací tabulky tranzitních RBridge byla odvozena od počtu RBridge v topologii, nikoliv podle počtu koncových stanic, kterých je typicky násobně více.

3.1.1 Jak to funguje

RBridge kombinuje funkci směrovače a přepínače. Všechny RBridge mezi sebou provozují instanci protokolu IS-IS, čímž si vyměňují informace o celé topologii. Protokol IS-IS se od verze provozované na směrovačích mírně liší a označuje se jako L2 IS-IS. L2 ovšem

neoznačuje úroveň oblasti, ale vrstvu v rámci modelu ISO/OSI. Klasické nasazení IS-IS jako směrovacího protokolu se označuje L3 IS-IS.

Samotné fungování protokolu se pak liší pouze minimálně. Používá *ploché* adresování - všechny RBridge patří do stejné *L1* oblasti a v *Ethernet* hlavičce je uveden přenášený protokol L2 IS-IS. Takové informace jsou zásadní pro správné generování nejkratších cest a distribučních stromů pro doručování rámců, jejichž destinace je neznámá, nebo se jedná o multicast, či broadcast.

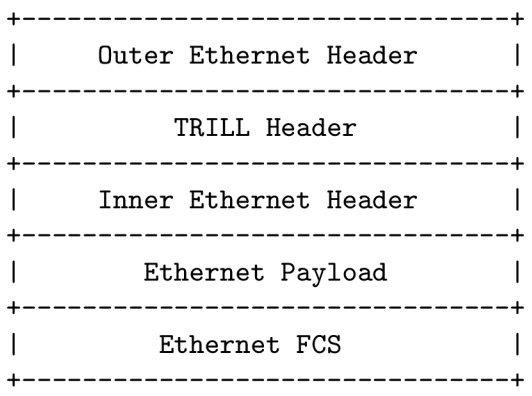
Pro zmírnění následků dočasných smyček provádí RBridge přeposílání na základě hlavičky s *Hop Count* – TRILL hlavička.

První RBridge, který přijme nativní rámec jej zapouzdří nejprve do TRILL hlavičky, určující výstupní RBridge a následně opatří ještě vnější Ethernet hlavičkou a přeposílá na další RBridge na cestě k požadované destinaci na základě směrovací tabulky vytvořené na základě informací vyměněných v rámci instance protokolu IS-IS. Formát zapouzdření můžeme vidět na obrázku 3.1. RBridge, který má danou koncovou stanici přímo připojenou, nebo zakončuje TRILL topologii směrem k dané koncové stanici, takový rámec rozbalí do jeho původní podoby a zašle na linku směrem k této stanici.

Z tohoto pohledu pracují RBridge obdobně jako směrovače, ale namísto IP hlavičky, přidávají TRILL hlavičku. TRILL hlavička také specifikuje zdroj a cíl (v případě *multi-destination* pak zdroj a distribuční strom) a tyto položky se po cestě nemění. Takto obalený rámec se opatří ještě další Ethernet hlavičkou s cílovou MAC adresou next-hop RBridge na cestě k cíli. Stejně jako IP router mění položky Ethernet rámce, ale zachovává adresu odesílatele i příjemce IP paketu[1], RBridge mění zdroj a cíl pouze ve vnější Ethernet hlavičce.

Multicast a broadcast je souhrně označen jako *multi-destination* provoz.

Protože unicast i multi-destination provoz v rámci RBridge campusu může využívat *multipathing*, je žádoucí vytvářet *full-mesh* topologie, na kterých je podstatně více poznat výhoda oproti stromové topologii.



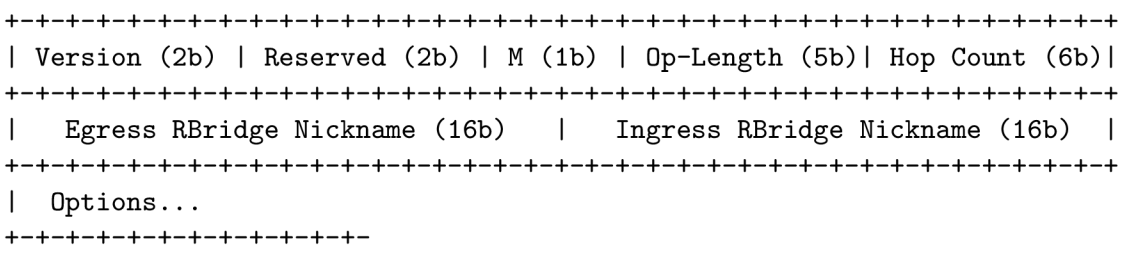
Obrázek 3.1: Formát zapouzdření rámce mezi dvěma RBridge

3.2 TRILL hlavička

Formát TRILL hlavičky vidíme na obrázku 3.2.

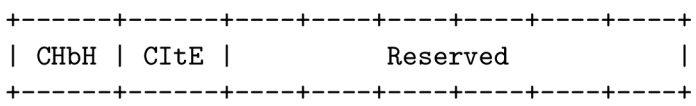
Význam jednotlivých polí je následovný:

[1] Neuvažujeme použití NAT a podobných technik.



Obrázek 3.2: Formát zapouzdření rámce mezi dvěma RBridge

- **Version** - dvou bitové číslo udávající používanou verzi. Aktuální verze je 0. RBridge musí ověřit, zda danou verzi podporuje. V opačném případě rámec v tichosti zahodí.
- **Reserved** - dva bity rezervované pro budoucí využití.
- **M** (Multi Destination) - indikuje, zda má být rámec doručen skupině příjemců dané třídou v distribučním stromě.
- **Op-Length** - udává délku volitelné části hlavičky v násobcích 4 B.
- **Hop Count** - obdoba TTL. RBridge zahazuje rámce s Hop Count 0. Tuto hodnotu nastavuje *Ingess RBridge* a každý další RBridge musí tuto hodnotu snížit alespoň o 1.
- **Egress RBridge Nickname** - dvou bytová dynamicky přidělená hodnota, která slouží jako zkratka IS-IS ID pro daný RBridge. Egress určuje RBrigde, kerým má daný rámec opustit TRILL oblast.
- **Ingress RBridge Nickname** - je steně velká položka jako pro Egress a určuje vstupní RBridge. V případě, že daný RBridge má více než jedno *nickname*, měl by vždy použít stejné pro zapouzdření rámců se stejnou zdrojovou MAC adresou a VLANou.
- **Options** - velikost této položky je udána pomocí *Op-Length*. Formát viz 3.3. Pokud *Options* obsahuje nějaké kritické informace pro přeposílání (CHbH - Critical Hop by Hop) je tento bit nastaven na 1. V případě, že některý RBridge po cestě takovou (kritickou) položku nepodporuje, rámec se zahodí.



Obrázek 3.3: Formát položky Options v TRILL hlavičce

3.3 Nickname

Nickname – *přezdívka* je 16 b dynamicky přiřazený identifikátor, který funguje jako zkratka *System-ID* a potencionálně jako označení více distribučních stromů se stejným kořenem.

Nickname s hodnotou `0x0000` je rezervovaná a vyjadřuje, že žádné *nickname* není nastavené. Hodnoty v rozsahu `0xFFC0` až `0xFFFF` jsou rezervované a nesmějí se používat.

Pokud má RBridge více *nickname*, měl by při přeposílání rámců se stejnou dvojicí zdrojová adresa, VLAN, použít jedno *nickname* jako `Ingress Nickname`.

3.3.1 Volba nickname

Volba probíhá pomocí protokolu IS-IS. Využívá se TLV #242 *Router CAPABILITY* rozšířené o sub-TLV #6 *Nickname*. Spolu s *přezdívkami* se přenáší také priorita jejich používání a priorita stát se kořenem stromu pro *multi-destination* provoz. RBridge může mít také ručně nakonfigurované *přezdívky*. Staticky konfigurované položky mají vždy vyšší prioritu nad všemi ostatními. Oznamovaná priorita je osmi bitová položka, kde nastavení nejvyššího bitu znamená, že *přezdívka* byla ručně konfigurována. Výchozí hodnota zbylých sedmi bitů je `0x40`.

V případě zjištění kolize dvou *přezdívek* si ji zachová RBridge s vyšší prioritou pro použití dané *přezdívky*. Pokud jsou obě priority stejné, porovnají se *System-ID*. RBridge s numericky vyšším *System-ID* vyhrává volbu a druhý RBridge si musí vygenerovat novou.

Generování probíhá náhodně z rozsahu volných *přezdívek*, na základě informací z LSP databáze. Při generování mohou být použity položky jako *System-ID*, čas, datum a další zdroje entropie dle [8]. Náhodný algoritmus by měl uniformně generovat hodnoty z dostupného rozsahu.

3.4 DIS, DRB a Appointed Forwarder

Stejně jako ISs v IS-IS i RBridges provádějí na multi access segmentech v rámci IS-IS instance volbu Designated IS, respektive Designated RBridge. Termíny Designated IS (DIS) a Designated RBridge (DRB) mají stejný význam. Pouze přenášejí již známý termín v rámci IS-IS do prostředí TRILLu a RBridge. Pokud hovoříme o DIS, máme na mysli L3-IS-IS. Pokud zmiňujeme DRB, jedná se o L2-IS-IS. Ve většině případů je záměna L2 a L3 IS-IS nepatrná. Termín DRB je pak použit v případech, kdy se chování L2 a L3 IS-IS liší. Pokud danou část chování obě verze sdílejí, může být pro označení činnosti DRB použita zkratka DIS, ale nikoliv naopak. Pro funkčnost čistě L2-IS-IS nebude použit termín DIS.

Volba DRB oproti volbě DIS se vyznačuje několika odlišnostmi a probíhá podle jiného schématu. Ačkoliv to specifikace `L3-IS-IS` nikde explicitně nezdůrazňuje, může za jistých okolností docházet k volbě několika DIS na jednom sdíleném segmentu. Za jakých okolností k tomuto jevu může docházet a jakým způsobem to řeší `L2-IS-IS` si řekneme v sekci 3.5.

Vítězný RBridge určí sebe, nebo některý jiný RBridge jako *Appointed Forwarder* pro všechny povolené VLANy na dané lince a informuje o tom pomocí TRILL Hello. Zvolení *Appointed Forwarder* pro VLAN, která není v rozsahu povolených VLAN, nemá žádný efekt.

Appointed Forwarder (AF) vstupní a výstupní brána sdíleného segmentu. Pouze *Appointed Forwarder* přijímá a odesílá nativní provoz za a na danou linku. Pokud nativní rámec příjme jiný RBridge, který není *Appointed Forwarder*, rámec zahodí. V takovém případě, je sdílený segmentem zajištěno doručení i pro *Appointed Forwarder*, který daný rámec zpracuje. RBridge je implicitně AF na všech svých portech, které nemají nastavený *end-station disabled* bit, nebo *P2P Hellos* bit.

DRB dále určuje *Designated VLAN*, která se bude na dané lince používat. Přes *Designated VLAN* proudí komunikace mezi RBridge a všechny TRILL-zapouzdřené zprávy včetně

ESADI protokolu a TRILL IS-IS rámců. Výjimku tvoří některé Hello zprávy, podrobněji v následující části.

3.5 TRILL Hello protokol

TRILL Hello protokol zavádí nový typ IS-IS zprávy a liší se od LAN Hello protokolu v rámci `L3-IS-IS`. Nový typ zpráv se označuje jednoduše jako `TRILL-Hello`.

Důvodem zavedení odlišného Hello protokolu a nového typu zpráv je, že v původním LAN Hello protokolu mohlo dojít ke zvolení více DIS na sdíleném segmentu. K tomuto jevu dochází v případech, kdy dva IS mezi sebou nemají obousměrnou konektivitu - navázané sousedství. Další možná příčina je, že LAN Hello používá uměle zvětšování Hello zpráv k testování MTU, a znemožnění komunikace mezi IS, které nepodporují danou velikost MTU. Na lince, která nedokáže přenášet uměle zvětšená Hello, nedojde k jejich úspěšnému přenosu, a tudíž druhé straně se jeví jako, že se na dané lince žádné Hello zprávy nevyskytují. To může vyústit k volbě několika pseudonodů na jednom sdíleném segmentu. V případě `L3-IS-IS` se takového chování nemusíme obávat, Takové chování je v pořádku pro třetí vrstvu (L3), ale nikoliv pro druhou (L2), kde delegování několika DRB může vést k vytvoření smyček.

Další důvod zavedení nového typu zpráv je pro umožnění, aby se mohla i podmnožina informací objevit v kterékoliv zprávě a byla řádně zpracována v duchu LSP a CSNP.

TRILL Hello protokol ovlivňuje volbu DRB. Oproti `L3-IS-IS` probíhá volba DRB pouze na základě priority a MAC adresy. RBridge při rozhodování nekontroluje, zda jej druhý RBridge v TRILL Hello uvádí jako dostupný. Jinými slovy RB1 nemusí splňovat *two-way check*, aby mohl být součástí rozhodování. Může se stát, že kvůli nevhodnému MTU v jednom směru, nebo při jednosměrném propojení, se DRB stane RB1, ale ostatní RBridge, které nemají s RB1 obousměrnou konektivitu jej neuvádí ve svých LSP. Stejně tak RB1 negeneruje ve jménu pseudonodu LSP položky pro uzly s kterými nemá obousměrnou konektivitu. [2]

Aby nevznikaly různé kliky z důvodu špatné konfigurace *Designated VLAN* posílají se TRILL Hello na všechny povolené VLAN na portu podle pseudokódu viz kód 3.1.

Kód 3.1: Pseudokód určení počtu generovaných TRILL Hello zpráv dle RFC 6325.

```
1  If sender is DRB
2    intersection ( Enabled VLANs,
3    union ( Designated VLAN, Announcing VLANs ) )
4
5  If sender is not DRB
6    intersection ( Enabled VLANs,
7    union ( Designated VLAN,
8    intersection ( Forwarding VLANs, Announcing VLANs ) ) )
```

Počet rozesílaných Hello zpráv lze minimalizovat nastavením *Announcing VLANs* na prázdnou množinu. Naopak pro maximalizování můžeme nastavit *Announcing VLANs* na *Enabled VLANs*, cože je výchozí chování protokolu.

[2] Tady by neuškodil obrázek s jednoduchou topologií a naznačenou jednosměrnou dostupností.

3.6 TRILL Hello

Formát TRILL Hello vychází z LAN Hello viz obrázek 2.3, pouze položka *Circuit Type* má vždy hodnotu `L1`, protože aktuální standard definuje `L2-IS-IS` pouze jako *Level 1.*. Jako všechny IS-IS zprávy začíná IS-IS obecnou hlavičkou viz obrázek 2.2. Velikost TRILL Hello zprávy, včetně vnitřní i vnější Ethernet hlavičky, by neměla přesáhnout 1470 B. Oproti Hello zprávám používaným v `L3-IS-IS` by TRILL Hello neměly být uměle zvětšovány pro testování MTU dané linky. Kontrolu MTU si podrobněji probereme v sekci 3.8. Jsou definovány následující položky, které jsou povinné v každém TRILL Hello:

- *Designated VLAN ID*,
- kopie vnějšího VLAN ID, kterým byla daná Hello zpráva označena při odeslání (slouží k detekci mapování),
- 16-ti bitový identifikátor, který v rámci daného zařízení jednoznačně určuje port, na kterém byla daná Hello zpráva odeslána,
- *nickname* odesílajícího RBridge,
- příznak pro identifikaci detekce VLAN mapování a
- příznak, který udává, že odesílající RBridge věří, že je *Appointed Forwarder* pro danou linku.

Všechny uvedené údaje jsou součástí TLV #143 *Multi-Topology-aware Port Capability* a jeho sub-TLV #1 *Special VLANS and Flags Sub-TLV*.

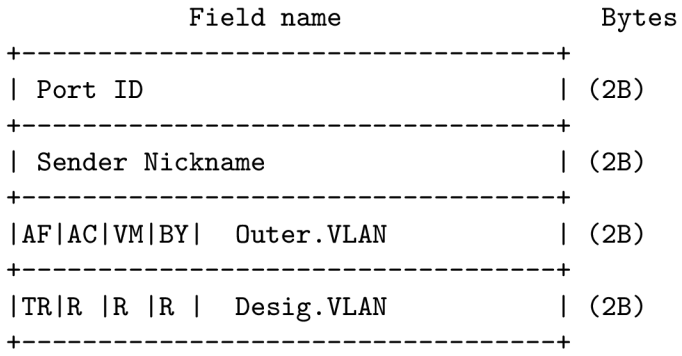
Multi-Topology-Aware Port Capability TLV #143 se využívá v TRILL Hello pro přenášení parametrů portů. Může se vyskytovat opakovaně. Jeho formát je na obrázku 3.4.

```
            Field name                  Bytes
+----------------------------------+
|R|R|R|R| Topology Identifier       | (1B)
+----------------------------------+
| sub-TLVs                         | (variable)
+----------------------------------+
```

Obrázek 3.4: Formát TLV #10 Multi-Topology-Aware Port Capability

- **R** jsou čtyři bity rezervované pro budoucí využití, které se přenášejí jako 0 a při přijetí jsou ignorovány.
- **Topology Identifier** je 12 b identifikátor ohlašované topologie. Pokud je nastaven na nulu, značí, že přenáší údaje o základní topologii.
- **sub-TLVs** může obsahovat různé sub-TLV a jeho délka je závislá na použitých sub-TLV. Možná sub-TLV jsou specifikována v sekci níže.

Special VLANs and Flags sub-TLV #1 je přenášeno v každé TRILL Hello zprávě. Jeho formát je znázorněn na obrázku 3.5. Má pevnou délku 8 B, ale pro nadřazené TLV se udává délka 10 B (2 B hlavička).



Obrázek 3.5: Formát sub-TLV #1 Special VLANs and Flags

- **Port ID** je 2 B označení portu, na kterém bude toto sub-TLV odesláno. Označení portu musí být jedinečný identifikátor v rámci daného RBridge.
- **Sender Nickname** obsahuje jedno z *nickname* patřící odesílacímu RBridge. Pokud žádné *nickname* nemá, posílá se pole nastavené na samé nuly.
- **Outer.VLAN** je kopie 12 b VLAN tagu z vnější *Ethernet* hlavičky s kterým byla TRILL Hello zpráva obsahující toto sub-TLV odeslána. Používá se pro detekci VLAN mapování.
- **Desig.VLAN** je identifikátor používané *Designated VLAN* na dané lince.

  AF, **AC**, **VM**, **BY** a **TR** jsou příznaky, které pokud jsou nastaveny na hodnotu 1 mají následující význam:

- **AF** – odesílací IS věří, že je *Appointed Forwarder* pro uvedené VLAN a port.
- **AC** – zdrojový port je nastaven jako *access*, takže je na něm zakázán TRILL provoz.
- **VM** – bylo detekováno VLAN mapování na této lince.
- **BY** – na tomto segmentu negeneruj pseudonode.
- **TR** – zdrojový port je nastaven jako *trunk*, čímž je zakázána služba koncových stanic (nativní rámce).

Další položky, které se mohou objevit v kterékoliv TRILL Hello zprávě zahrnují:

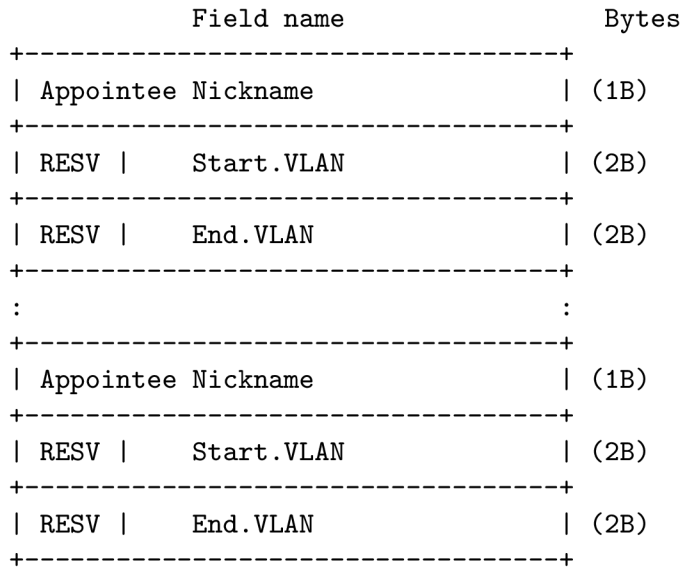
- množina VLAN, na kterých jsou povoleny koncové stanice,
- příznak indikující, že daný port je nakonfigurován jako
  1. *access*
  2. *trunk*
  3. *bypass pseudonode*
- seznam *Appointed Forwarders* pro danou linku a uvedené VLAN a
- seznam TRILL sousedů.

Množina VLAN s povolenou *end-station service* je přenášena pomocí TLV #143 sub-TLV #2 *Enabled-VLANs Sub-TLV*.

U sady příznaků je udivující, že nejsou povinné ve všech TRILL Hello, protože jsou součástí TLV #143 sub-TLV #1, stejně jako například *nickname* a identifikátor portu, které povinné jsou. Pevný formát tohoto sub-TLV nás stejně nutí všechny informace uvést v každé TRILL Hello zprávě. Nastavení *bypass pseudonode* způsobí, že se pro danou linku negeneruje *pseudonode* a sousedství se reportuje jako v případě point-to-point. Zároveň se tento příznak nastavuje pouze v případě, že daný RBridge od svého zapnutí neviděl současně dvě a více sousedství na dané lince. Nastavení tohoto příznaku neovlivňuje způsob zasílání LSP, ale pouze jejich generování.

Pro seznam *Appointed Forwarders* je opět je použito TLV #143, ale se sub-TLV #3 *Appointed Forwarders Sub-TLV*.

Pomocí *Appointed Forwarders* sub-TLV #3 DRB oznamuje ostatní IS o tom, koho zvolil jako *Appointed Forwarder* pro jeden, či více rozsahů VLAN ID. Oznamovaní *Appointed Forwarders* jsou platní na lince, na které je dané TRILL Hello přenášeno. Jeho formát je znázorněn na obrázku 3.6. Jeden záznam má délku 6 B, takže délka nabývá hodnot násobků 6.

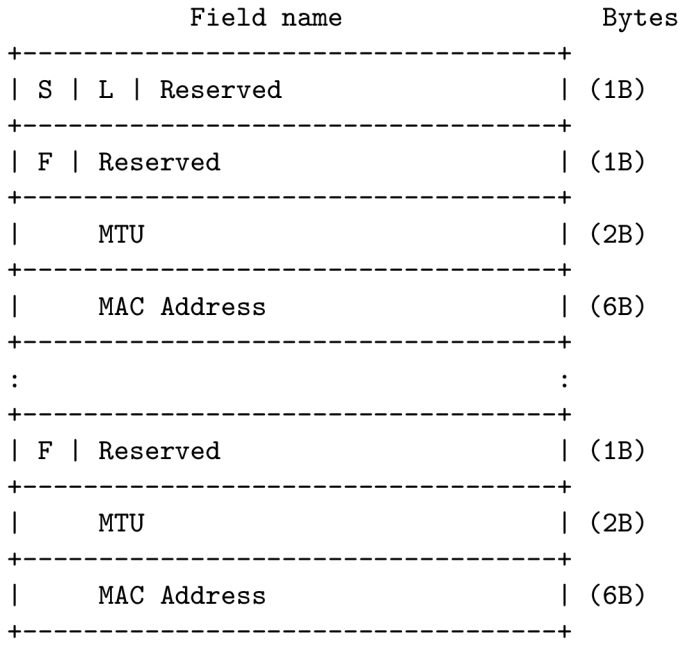


Obrázek 3.6: Formát sub-TLV #3 Appointed Forwarders

- **Appointee Nickname** je 2 B označení, *nickname*, IS, který byl zvolen jako *appointed forwarder* pro rozsah VLAN ID od *Start.VLAN* po *End.VLAN*.
- **Start.VLAN** a **End.VLAN** určují rozsah platnosti *Appointee Nickname*. Rozsah je včetně uvedených VLAN ID.
- **RESV** jsou 4 b rezervované pro budoucí využití.

A konečně seznam TRILL sousedů, ten má definované vlastní TLV #145 *TRILL Neighbor TLV*. Využívá podobného mechanismu jako CSNP.

TRILL Neighbor TLV #145 se používá v TRILL Hello místo *IIS Neighbor* TLV #6. Jeho struktura je uvedena na obrázku 3.7.



Obrázek 3.7: Formát TLV #145 TRILL Neighbor

- **S** – *Smalllest flag*, pokud je nastaven, tak množina oznamovaných sousedů obsahuje souseda s nejmenší MAC adresou.
- **L** – *Largest flag*, pokud je nastaven, tak množina oznamovaných sousedů obsahuje souseda s největší MAC adresou.
- **F** je příznak informující o selhání MTU testu, pro požadované MTU v rámci celého *campusu*, k oznamovanému sousedovi. Podrobnosti o testování MTU viz sekce 3.8.
- **Start.VLAN** a **End.VLAN** určují rozsah platnosti *Appointee Nickname*. Rozsah je včetně uvedených VLAN ID.
- **MTU** je největší úspěšně otestované MTU s daným sousedem, nebo nula, pokud nebyl proveden MTU test.
- **MAC Address** je adresa souseda stejně jako *LAN Address* v TLV #6 viz sekce C.5.

Pokud se všechny MAC adresy vejdou do jednoho TLV, budou oba bity S i L nastaveny na jedničku. Jestliže se všechny záznamy o sousedech nevejdou do jednoho TLV, tak se nejvyšší MAC adresa, která bude v TLV s nastaveným *Smallest flag*, musí objevit v nějakém dalším TRILL Neighbor TLV. Stejně tak nejnižší MAC adresa, která bude v TLV s nastaveným *Largest flag*, musí být uvedena v dalším TLV, které může být součástí jiného TRILL Hello. Za určitou dobu se musí vyskytnout celý rozsah MAC adres sousedů. Jednotlivé záznamy musí být vzestupně seřazeny podle MAC adresy.

Pokud daný daný IS věří, že nemá žádné sousedy, musí zaslat TRILL Neighbor TLV prázdné.

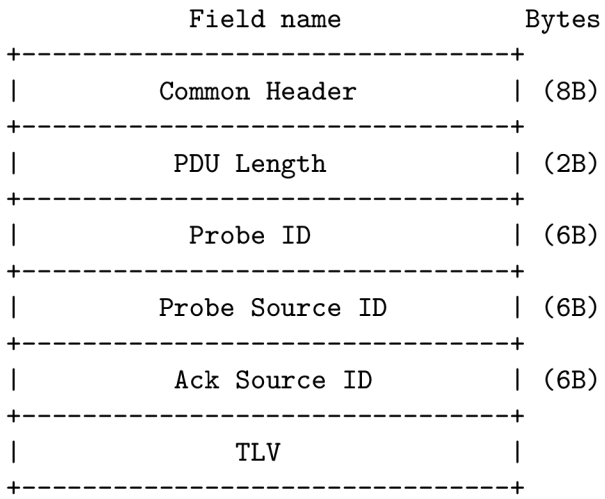
Celková délka je $1+9*N$, kde N je počet oznamovaných sousedů, který může být i nula. Jeden záznam tvoří položky F, 7 b *Reserved*, *MTU* a *MAC Address*.

3.7 Zjištění MTU

Všechny RBridge v campusu musí podporovat určité minimální MTU, aby bylo zajištěno správné doručování IS-IS zpráv. Výchozí hodnota je 1470 B a je definována v původní verzi IS-IS. Aby mohl IS-IS spolehlivě pracovat i v prostředí s nižším MTU, případně, aby dokázal naplno využít možnosti větších MTU, každý RBridge, volitelně, zasílá pomocí TLV #14 největší podporovanou velikost zpráv. V případě, že toto TLV IS nerozesílá, předpokládá se implicitní hodnota 1470 B. Nejmenší hodnota deklarovaná některým RBridge se použije jako *campus-wide MTU*.

3.8 Kontrola MTU

Na rozdíl od LAN Hello, i PTP Hello v `L3-IS-IS`, neprobíhá kontrola minimální hodnoty MTU pomocí uměle zvětšování Hello zpráv. `L2-IS-IS` na to používá dva nové typy zpráv přidaných do IS-IS. Jejich formát je totožný a můžete ho vidět na obrázku 3.8. Liší se pouze položkou *PDU type*. Pro *MTU Probe* je vyhrazen kód `23` a pro *MTU Ack* kód `28`. Zmíněné *MTU PDU* slouží k ověření MTU pouze na *Designated VLAN*.



Obrázek 3.8: Formát MTU PDU

- **PDU Length** určuje délku zprávy včetně obecné hlavičky.
- **Probe ID** nastavuje původce *MTU Probe* na hodnotu, podle které identifikuje odpověď na tento dotaz.
- **Probe Source ID** opět nastavuje původce *MTU Probe* a to na svoje *System-ID*. Odpovídající IS jej pouze zkopíruje do *MTU Ack*.
- **Ack Source ID** nechává odesílatel *MTU Probe* prázdné (nastaví na samé 0). Odesílatel *MTU Ack* vyplní svým *System-ID*.
- **TLV** může obsahovat *Authentication TLV* a musí být doplněna na požadovanou délku pomocí *Padding TLV*. V případě, že by velikost *MTU PDU* včetně obecné hlavičky byla pouze o 1 B menší než testovaná délka, nemohli bychom požadované

velikosti dosáhnout, protože minimální velikost *Padding TLV* jsou 2 B. Ale vzhledem k minimální velikosti *MTU PDU* a k tomu, že minimální MTU na 802.3 linkách je 1470 B by tato situace neměla nastat.

Zasílání *MTU Probe* zpráv je nepovinné, ale pokud RBridge takovou zprávu obrdží, musí na ni odpovědět pomocí *MTU Ack* s odpovídající velikostí.

MTU Probe může být multicast na adresu `All-RBridges`, nebo na konkrétní adresu dotazovaného RBridge. *MTU Ack* se většinou posílá jako unicast na dotazující RBridge, ale může být zaslána i jako multicast na `All-RBridges`.

Pokud dotazující RBridge nedostane pro *MTU Probe* s velikostí X odpověď po k dotazech[3], předpokládá, že daná linka nepodporuje danou velikost zpráv. Pokud je X menší než dohodnuté minimální MTU pro celý campus, nastaví odpovídající bit ve svých Hello zprávách.

3.9 Detekce VLAN mapování

Detekce VLAN mapování je důležitá, protože za jistých okolností může způsobovat vytvoření smyčky. Každá TRILL Hello zpráva obsahuje Outer.VLAN ID s kterým byla odeslána. Pokud dojde ke změně vnějšího VLAN ID, přijímající RBridge to zjistí porovnáním Outer.VLAN ID z Ethernet hlavičky a Outer.VLAN ID z TRILL Hello. Pokud DRB detekuji VLAN mapování, znovu určí *Appointed Forwarders* tak, aby pro všechny mapované VLAN, byl pouze jeden *Appointed Forwarder*.

3.10 Distribuční stromy

Pro doručování multicast a broadcat provozu (multi-destination), používá TRILL *distribuční stromy*. Všechna propojení v rámci distribučního stromu jsou obousměrná. Ačkoliv jediný strom je dostatečný pro celý campus, jsou generovány i další stromy pro umožnění *multipathing* pro multi-destination rámce a možnost zajištění volby kořene distribučního stromu blízkého, nebo stejného s *Ingress RBridge*.

Další míra flexibility je zajištěna možností RBridge získat více *nickname*, a díky tomu být kořenem několika distribučních stromů.

RBridge si předem generuje všechny distribuční stromy, které by mohl použít. Všechny RBridge potřebují znát:

- kolik stromů je potřeba vytvářet,
- které konkrétní stromy se musí generovat,
- přiřazené číslo konkrétnímu stromu a
- které stromy každý Ingress RBridge může použít (pro vytváření *Reverse Path Forwarding* filtrů).

Každý RBridge rozesílá ve svých LSP prioritu pro každý svůj *nickname*, aby se stal *tree root*. Jedná se o 16 b číslo s výchozí hodnotou `0x8000`. Strom s numericky vyšší hodnotou má vyšší prioritu. V případě shody jsou další kritéria *System-ID* a *nickname*. Opět bereme numericky vyšší hodnotu jako vyšší prioritu.

[3] k je konfigurovatelný parametr. Výchozí hodnota je 3.

Všechny RBridge také pomocí LSP informují o maximálním počtu stromů, které jsou schopné spočítat a počet stromů, které požadují po ostatních RBridge aby generovaly. Počet generovaných stromů v rámci campusu, k je počet požadovaných stromů RBridge s nejvyšší *tree-root* prioritou *RB1*, ale ne více než nejmenší počet stromů, které je schopen některý z RBridge v rámci capmusu generovat. Pokud *RB1* přesně neurčí, které stromy požaduje, aby se generovaly, vybere k stromů s nejvyšší prioritou. V případě, že *RB1* explicitně ve svých LSP určí s stromů, které chce, aby byly generovány, pak je vybráno prvních k.

Obdobně jako u původní specifikace IS-IS, pokud některý RBridge udává, že počet stromů, které může generovat, nebo, které chce, aby byly generovány je nula, aplikuje se výchozí hodnota. Výchozí hodnota pro počet generovaných stromů je jedna.

3.11 Přechod na RBridge

Nejlepší způsob nasazení RBridge je kompletní výměna stávajících přepínačů. To ovšem v reálném nasazení často není možné a musí být zajištěna doba, po kterou spolu budou oba typy zařízení spolupracovat. Předpokládá se, že typické nasazení bude probíhat postupně. Proto jsou RBridge navrženy v souladu s *802.1 customer bridges*.

3.12 Učení koncových stanic

RBridge se musí naučit trojici MAC adresa, VLAN a port pro všechny koncové stanice přímo připojené na linkách, kde pro danou VLAN jednají jako AF. Pro vzdálené koncové stanice se navíc učí trojici MAC adresa, VLAN a RBridge, který má danou stanici přímo připojenou.

Je celkem pět způsobů jak se může RBridge naučit adresu koncové stanice:

1. z rozboru rámců na VLAN, pro kterou je AF (zdrojová MAC, VLAN, port),
2. z nativních rámců, které rozbaluje (zdrojová MAC, VLAN, ingress RBridge nickname),
3. pomocí *Layer 2* registračních protokolů (zdrojová MAC, VLAN, port),
4. pomocí ESADI protokolu viz sekce 3.14 a
5. pomocí ruční konfigurace.

První dva způsoby jsou povinné pro všechny implementace RBridge, ale mohou být konfiguračně vypnuty pro konkrétní port, nebo VLAN. Způsob 3 a 4 jsou volitelné a poslední, pátý, je doporučený.

Ke každému výše zmíněnému záznamu je navíc přidělena položka *confidence*, udávající důvěryhodnost naučené informace. Záznamy naučené pomocí ESADI protokolu mají *confidence* v rozsahu 0 - 254. Ručně zadané hodnoty mají výchozí *důvěryhodnost* `255`, ale může být změněna na jinou hodnotu. Pro záznamy naučené pomocí prvních tří způsobů je výchozí *důvěryhodnost* `0x20`.

Všechny záznamy mají časovač, po jehož vypršení jsou vymazány. Pro existující záznamy je časovač vyresetován pouze pokud je naučen daný záznam se stejnou, nebo vyšší *důvěryhodností*.

Nastavení různé *důvěryhodnosti* pro každý způsob učení, umožňuje administrátorovi určité metody preferovat před ostatními.

3.13 Zapomínání koncových stanic

Stejně důležité jako učení se koncových stanic je jejich zapomínání. Jedním ze způsobů je vypršení časovače označeného jako *Aging Time*. Identifikuje interval mezi dvěma událostmi učení, ať už na základě rozboru lokálních rámců, nebo rozbalení nativního provozu. *Aging Time* je v rozsahu od 10 do 1000000 sekund a ve výchozím stavu nastaven na 300 s.

Dalším způsobem je zrušení RBridge jako AF pro danou VLAN. Všechny záznamy s takovou VLAN naučeny na základě rozboru nativních rámců jsou odstraněny.

Pokud provozuje ESADI protokol, přestane dané záznamy propagovat pomocí svých LSP, ale ještě odešle LSP nulující všechny takové záznamy.

3.14 ESADI

End Station Address Distribution Information je protokol pro distribuci informací o *koncových stanicích*. Pro urychlení přechodu mezi *známým* a *neznámým* unicast provozem je volitelně mezi všemi *Appointed Forwarders* provozován protokol ESADI. Všechny ESADI zprávy mají formát IS-IS rámců, ale jsou navíc ještě zapouzdřeny v TRILL a vnější Ethernet hlavičce a tváří se tak jako TRILL Data rámce viz obrázek 3.9. RBridge, který neprovozuje TRILL ESADI protokol, nebo není AF pro danou VLAN, takový rámec nezpracovávají, a pouze přeposílají dál. Všechny přestupní RBridge (ne-AP) takové rámce přeposílají jako obyčejný multi-destination provoz. Díky tomuto přeposílání se zdá, že všichni RBridge provozující ESADI protokol, jsou součástí jednoho virtuálního okruhu pro danou VLAN.

Vnější cílová MAC adresa `Outer.MacDA` je nastavena na adresu `All-RBridges` a vnější VLAN `Outer.VLAN` na identifikátor *Designated VLAN* pro danou linku.

3.14.1 TRILL ESADI informace

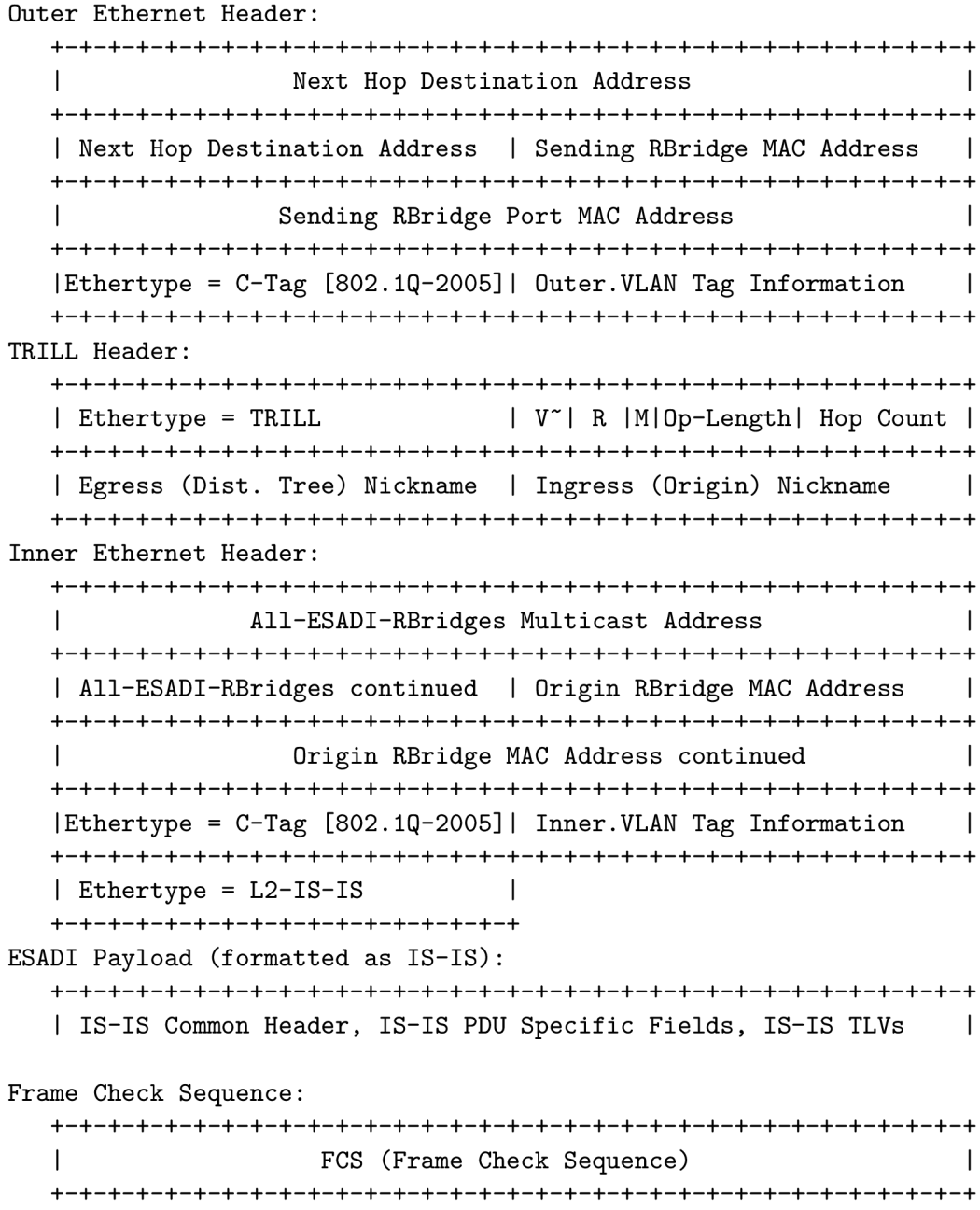
Informace rozesílané TRILL ESADI protokolem je seznam známých MAC adres koncových stanic připojených do zdrojového RBridge. Pro každou adresu je uvedena míra důvěry *confidence* v rozsahu $0 - 254$ viz sekce 3.12.

Každý AF tak při přijetí nativního rámce určenému vzdálenému cíli, může odeslat jako unicast TRILL-zapouzdřený rámec na specifikovaný vzdálený RBridge, aniž by někdy zpracovával jiný provoz s adresou cílové stanice. Za předpokladu, že daná stanice v nedávné době generovala provoz přes kterýkoliv z RBridge v campusu. Jednotlivé RBridge tak nemusí rozesílat unicast provoz jako multi-destination.

3.15 Rozdělení provozu

RBridge dělí provoz na pět základních kategorií.

1. Layer 2 control rámce (např. Bridge PDU (BPDU)),
2. nativní rámce (*non-TRILL-encapsulated* rámce),
3. *TRILL Data* rámce (*TRILL-encapsulated* rámce),
4. *TRILL control* rámce a
5. *TRILL other*



Obrázek 3.9: Formát TRILL ESADI rámce

3.15.1 Layer 2 control rámce

Layer 2 control rámce mají multicastovou cílovou adresu v rozsahu `01-80-C2-00-00-00` až `01-80-C2-00-00-0F`, nebo rovnu `01-80-C2-00-00-21`.

3.15.2 Nativní rámce

Nativní rámce jsou takové rámce, které nespadají pod *Layer 2 control* rámce a mají *Ethertype* jiný než `TRILL`, nebo `L2-IS-IS` a zároveň cílová MAC adresa nepatří do rozsahu

16 multicast adres rezervovaných pro TRILL (`01-80-C2-00-00-40` až `01-80-C2-00-00-4F`).

3.15.3 TRILL Data rámce

TRILL Data rámce mají *Ethertype* `TRILL` a pokud se jedná o multicast mají cílovou MAC adresu nastavenu na `All-RBridges`.

3.15.4 TRILL control rámce

TRILL control rámce mají *Ethertype* `L2-IS-IS` a pokud se jedná o multicast mají cílovou MAC adresu nastavenu na `All-IS-IS-RBridges`.

3.15.5 TRILL other rámce

TRILL other rámce mají jednu z 16-ti registrovaných MAC adres pro TRILL a zároveň jinou než `All-RBridges` a `All-IS-IS-RBridges`. Takto klasifikované rámce jsou zahozeny.

3.16 Zpracování rámců

Podrobný popis zpracování všech typů rámců je uveden viz [22, 4.6]. Zde si popíšeme pouze hlavní principy.

3.16.1 Nativní

Pokud je cílová adresa unicastová, prohledá se MAC tabulka pro nalezení výstupního RBridge. Rámec může být:

- zpracován lokálně,
- poslán v nativní podobě na lokální port,
- TRILL zapouzdřený odeslán na vzdálený RBridge,
- zpracován jako *multi-destination*, pokud je cílová adresa neznámá.

Multi-destination rámec je zaslán v nativní podobě na všechna rozhraní, kde je *Appointed Forwarder* a TRILL zapouzdřený všem RBridge sousedům podle zvoleného distribučního stromu.

3.16.2 TRILL Data

Zpracování se dále děli podle toho, jestli má rámec nastaven `M` příznak v TRILL hlavičce.

Unicast provoz se přeposílá na *next-hop* ve směru k cílovému RBridge. V TRILL hlavičce se sníží hodnota `hop count`.

Zpracování *multi-destination* rámce je obdobné jako u nativního provozu. Kopie bez TRILL hlavičky se pošle v nativní podobě na všechna rozhraní, kde je *Appointed Forwarder* pro danou VLAN specifikovanou ve vnitřní *Ethernet* hlavičce a TRILL zapouzdřený všem RBridge sousedům podle zvoleného distribučního stromu.

3.16.3 TRILL control

TRILL Control rámce zpracovává protokol IS-IS.

3.16.4 TRILL other

Rámce označené jako TRILL other jsou zahazovány bez zasílání potvrzení o přijetí.

3.17 Podpora na Cisco zařízeních

Aktuálně není TRILL jako takový podporován na žádném Cisco zařízení. Cisco ovšem vyvíjí vlastní proprietární protokol FabricPath[2], který dokáže běžet v kompatibilním režimu s TRILL. FabricPath je dostupný na přepínačích Nexus 7000.

Kapitola 4

Implementace protokolů

V této části si uvedeme hlavní principy protokolu IS-IS, tak jak byly implementovány, případně zmíněny odchylky od specifikace dle ISO 10589:2002. Pro podrobný popis zdrojových kódů, prosím, prostudujte přiloženou programovou dokumentaci. Zdrojové kódy se nacházejí na přiloženém CD a v GIT repozitáři projektu ANSA.

Původní návrh implementace počítal pouze s dokončením stávajícího protokolu IS-IS, ve kterém bylo implementováno navazování sousedství a výměna LSP na *L1*. Zbývalo už tedy jenom zkopírovat jednotlivé metody pro zprovoznění *L2* a implementovat SPF pro nalezení nejkratších cest a následné propagování těchto informací do směrovací tabulky. Bohužel se ukázalo, že se bude muset přepracovat většina stávajících metod, protože v částech jako volba DIS a navazování sousedství byly chyby a nakládání s LSP bylo v rozporu s ISO 10589:2002 [13].

Proto bylo jako cíl stanoveno opravení stávající implementace IS-IS a případně vytvoření základu pro TRILL rozšíření. Tento cíl byl splněn a kromě vytvoření protokolu IS-IS včetně směrovací tabulky pro CLNS byl také implementován samotný TRILL. Celá práce je vytvořena v jazyku C++ do prostředí OMNeT++ a je součástí projektu ANSA.

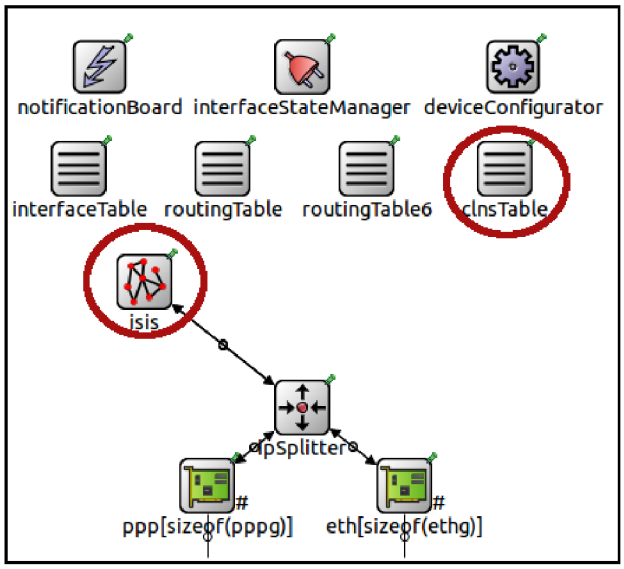
4.1 OMNeT++

OMNeT++ je simulační prostředí pro modelování nejrůznějších jevů. S rozšířením INET se z něj stává simulační nástroj pro modelování síťové komunikace. ++ v jeho názvu napovídá, že je vytvořen v jazyce C++. Základní prvek simulace tvoří modul. Jednotlivé moduly jsou pak spojovány pomocí NED souborů, čímž vytváří entity odpovídající reálným síťovým zařízením. Takto vytvořené entity i samotné moduly je možné mezi sebou propojovat pomocí bran.

4.2 IS-IS

Implementace protokolu IS-IS je součástí modelu směrovače `AnsaISISRouter`, jehož strukturu můžete vidět na obrázku 4.1. Červeně jsou zvýrazněny moduly, které jsme vytvářeli, nebo upravovali.

Celý protokol tvoří jeden modul reprezentovaný hlavní třídou `ISIS`. Kromě zpráv, které si vyměňují jednotlivé IS, je využívána třída `ISISTimer` pro hlídání vypršení časových limitů.



Obrázek 4.1: Strutura modelu vytvořeného směrovače.

Na obrázku 4.2 je diagram tříd souvisejících s IS-IS. Z uvedeného obrázku jsou odstraněny verze proměnných a metod pro *Level 2* a jednoduché `get` a `set` metody.

4.3 Konfigurace

Chování protokolu je možné ovlivnit pomocí konfigurovatelných proměnných. Jejich hodnota se odvozuje podle toho na jaké úrovni byly zadány. Každá úroveň má jinou prioritu. Nejnižší prioritu mají hodnoty definované v hlavičkovém souboru `ISISTypes.h` odkud jsou hodnoty přebírány, pokud nejsou zadány globálně pro celý IS-IS proces, ani pro jednotlivé rozhraní. Jedná se o výchozí hodnoty dle ISO 10589:2002, které tak eliminují nutnost zdlouhavého vytváření konfiguračního souboru pro jednoduché testování.

Druhou úroveň tvoří *globální* proměnné pro celý proces, které se zadávají v konfiguračním souboru ve formátu **XML**. Ty, pokud nejsou zadány, jsou přebírány právě z hlavičkového souboru `ISISTypes.h`.

Nejvyšší úroveň tvoří hodnoty zadané pro konkrétní rozhraní. Takové parametry přepíší výše zmíněné. Tento způsob je zvolen pro jednodušší tvorbu konfiguračních souborů a využívá obdobný přístup jako je tomu při dědění tříd v objektovém programování. Pokud chceme používat výchozí konfiguraci IS-IS, stačí zadat pouze NET adresu, z které se odvodí *System-ID* a *Area-ID*. Pro ostatní parametry se použijí výchozí hodnoty. Případně stačí změnu zadat na úrovni procesu a automaticky se aplikuje do nastavení všech rozhraní. Šablona konfiguračního souboru je uvedena v příloze D.1. Parametry, které se konfigurují na rozhraních, jsou uvozeny klíčovým slovem `ISIS`, abychom se vyhnuli možné kolizi s ostatními protokoly. Většina parametrů má svůj přímý ekvivalent v Cisco IOS i Juniper JUNOS, s ohledem na možnost jednoduchého importu reálné konfigurace.

Obrázek 4.2: Class diagram IS-IS

4.4 Plánování zpráv

Protože řízení simulace je v OMNeTu prováděno pomocí zasílání zpráv sám sobě, byla nutnost kontrolovat a měnit jednotlivé časy na všech místech implementace jednoduchým způsobem. Proto místo volání funkce `scheduleAt()` z různých míst, došlo k vytvoření centrálního plánování pomocí metody `ISIS::schedule`, která má dva parametry. První je povinný - `ISISTimer *timer` a druhý nepovinný - `double timee` čas. Pro případy, kdy plánujeme zprávu na základě vlastní konfigurace nám stačí pouze objekt `ISISTimer`, pomocí nějž vyhledáme požadované informace například pro dané rozhraní a úroveň. Pokud ale plánujeme například vypršení platnosti LSP, potřebujeme znát hodnotu z přijatého LSP. V takové situaci využijeme právě druhého volitelného parametru.

Další inovací je zavedení *jitter* 25 % při plánování většiny zpráv, aby došlo k rozložení nárazového množství zpráv. Tato vlastnost patří mezi vyžadované specifikací a je užitečná pro závěrečné simulování, ale při vývoji poněkud ztěžuje odlaďování.

4.5 Hello zprávy

Základ výměny Hello zpráv byl převzat z původní implementace. Navíc byl rozšířen o podporu point-to-point rozhraní. Nejedná se ovšem o opravdové point-to-point spojení, ale pouze emulaci na Ethernetu při přímém propojení dvou IS. Toto zjednodušení nenarušuje principy IS-IS, pouze ulehčuje implementační detaily.

Pro oba typy Hello zpráv (LAN a PtP) byl zaveden 3-way handshake pomocí TLV #240. Pro správné přiřazení Hello zprávy k vytvořenému sousedství (dále jako adjacency) je kromě *Source-ID*, z hlavičky konkrétní Hello zprávy, porovnána také odesílatelova MAC adresa a rozhraní, přes které byla zpráva přijata, s odpovídajícími informacemi pro dané adjacency. Díky tomu můžeme úspěšně odlišit několik redundantních spojení mezi dvěma IS.

Došlo také na změny pro volbu a rezignaci DISu na LAN. V původní verzi docházelo k nesprávnému vyhodnocení priority v případě, že dva (a více) IS měly nastavenu vyšší prioritu než zbývající IS na LAN. K ustanovení DIS nedochází při přijetí každé Hello zprávy, ale pouze, pokud je zjištěna změna při volbě DIS. Samotná volba/kontrola DIS samozřejmě i nadále probíhá při přijetí každé LAN Hello zprávy. Pokud nastane změna DIS, spustí se metoda `ISIS::purgeRemainLSP`, která zajistí *system wide purge* všech LSP předchozího DIS.

Další situací, kdy může dojít ke změně DIS, je při vypršení *Hold time* intervalu. Při jeho vypršení již nedochází k resetování DIS na všech rozhraních, ale pouze na rozhraní, které odpovídá dané adjacency, a to navíc pouze pokud byl daný IS na uvedeném rozhraní veden jako DIS.

Pro plánování Hello zpráv přibyly v konfiguraci parametry `ISIS-L1-Hello-Interval` a `ISIS-L1-Hello-Multiplier`. Oba parametry jsou *per-interface* a mají svůj ekvivalent pro *L2*. Můžeme je rovněž konfigurovat i v rámci celého procesu. V tom případě vypouštíme `ISIS` z názvu parametru. Slouží pro implementaci principu, který můžeme nalézt v Cisco IOS, a dává nám možnost zasílat Hello zprávy v intervalu kratším než 1 s. Výsledný *Hold time* interval, vypočítaný z výše zmíněných parametrů, platí pouze pro non-DIS rozhraní. Na rozhraních, kde daný IS pracuje jako DIS, se používá pouze třetina vypočítané hodnoty. Zajistíme tak rychlejší detekci výpadku DIS.

4.6 LSP

Zpracování LSP bylo předěláno kompletně tak, aby využívalo periodického rozesílání LSP na základě nastavených SRM a SSN příznaků a aby se rozesílaly pouze změněné LSP. Z toho důvodu musel být pozměněn i formát LSP databáze.

4.6.1 Generování

Pokud při vypršení intervalu pro generování LSP dojde k vygenerování stejného setu LSP, jejich *Sequence number* se nezmění. Ovšem oproti doporučení z ISO 10589:2002 nejsou jednotlivé adjacencies vázány na dané LSP. Stačí tak změna v jednom LSP (přidání/ubrání/změna parametru) a dojde k přepsání všech následujících LSP daného IS. Pojmem *následujících LSP* jsou myšleny všechny LSP s *Fragment-ID* stejným, nebo vyšším než první rozdílné LSP. Takové chování sice není v rozporu se specifikací, nicméně je navrženo vylepšení. Do struktury pro adjacency se navíc přidá reference na LSP a při opětovném generování LSP se bude kontrolovat, zda tyto informace byly naposledy v daném LSP. Dojde tak k zredukování množství zasílaných zpráv při drobné změně konfigurace daného IS.

Generování LSP navíc podporuje vytváření fragmentů. Pro možnost simulovat tuto vlastnost i na méně rozsáhlé konfiguraci slouží parametr `ISIS_LSP_MAX_SIZE`, kterým můžeme donutit IS-IS využívat fragmentaci i na menší zprávy.

Nově jsou do databáze ukládány celé LSP zprávy tak, jak byly přijaty. To usnadňuje jejich pozdější přeposílání a respektuje filozofii, že pokud IS některému TLV nerozumí, tak jej nezpracovává, ale posílá dál.

4.6.2 Periodické zasílání

Aby nemohlo dojít k *vyhladovění* LSP při čekání na přeposlání, je mechanismus SRM a SSN příznaků pro každé LSP navíc doplněn speciální frontou pro každou kombinaci úroveň (*L1*,*L2*)/příznaky (SRM, SSN)/typ rozhraní (broadcast, PtP). Zároveň tím respektujeme doporučení ve specifikaci, které umožní efektivnější způsob kontroly nastavení příznaků u jednotlivých LSP. Namísto kontroly všech příznaků pro všechny LSP v databázi, stačí zkontrolovat, jestli je daná fronta neprázdná. Fronty pro broadcast a PtP jsou odděleny proto, že na rozdíl od PtP, kde se posílají všechny LSP čekající ve frontě, na broadcast rozhraní se náhodně vybírá jedno z aktuálně čekajících LSP. Je to z důvodu ochrany zahlcení DIS, pokud by mu všechny ostatní IS poslali kompletní frontu v jeden okamžik.

Náhodný výběr navíc přispívá k tomu, aby DIS nedostával stejné LSP od všech ostatních IS na segmentu. Aby nedocházelo k zasílání stejného LSP několikrát v průběhu jedné iterace kontroly SRM příznaků, je nastavení SRM příznaku podmíněno jeho aktuálním stavem. Pokud je příznak již nastaven, nemůže být znovu přidán do fronty.

Dalším možným rozšířením je vytvořit odesílací frontu, která by zajišťovala rozestup mezi odesílanými LSP. Výchozí interval bývá 33 ms.

4.6.3 Refresh LSP

Refresh LSP probíhá každých `ISIS_LSP_SEND_INTERVAL` sekund, což je ve výchozím stavu 5 s. Opět je možné tento interval nastavit v konfiguraci pomocí `L1_LSP_Send_Interval`. Refresh interval se nastavuje odděleně pro *L1* a *L2*. Aktualizovány jsou pouze LSP patřící danému IS a všem jeho DIS inkarnacím. Aby bylo LSP obnoveno, musí být jeho *Remaining lifetime* větší než nula a přidružený časovač - `ISISTimer` - nesmí mít nastaven typ

`LSP_DELETE`. V takovém případě se jedná o LSP, pro které již bylo iniciováno odstranění a je uchovávána pouze jeho hlavička po dobu 2 × `ISIS_LSP_MAX_LIFETIME`. Po uplynutí i tohoto intervalu dojde k úplnému vymazání daného LSP z databáze.

4.6.4 Handle LSP

Při přijetí LSP je kontrolováno, zda existuje adjacency pro odesílající IS. Musí souhlasit nejenom *Source-ID*, ale i rozhraní, přes které byla zpráva přijata. V opačném případě je LSP zahozeno. Celé zpracování LSP při přijetí je v podstatě zjednodušeno pouze na nastavování příslušných SSN a SRM příznaků.

4.6.5 LSP Purge

LSP Purge pracuje ve dvou režimech podle toho, zda se jedná o volání smazání LSP z databáze při vypršení časovače a tedy položky *Remaining lifetime*, nebo IS obdržel LSP s *Remaining lifetime* rovný nule od svého souseda. Při vypršení časovače LSP, které je uložené v databázi, nedochází k jeho úplnému vymazání. V databázi se i nadále uchovává hlavička, ale celá TLV část je odstraněna. Typ přidruženého časovače se změní na `LSP_DELETE` a signalizuje, že pokud vyprší, má se dané LSP úplně vymazat. Varianty `purgeRemainLSP` a `purgeMyLSPs` slouží k iniciaci smazání série LSP.

4.7 Sequence Numbers PDU

4.7.1 CSNP

CSNP zprávy jsou plánovány pro každé rozhraní zvlášť dle nastaveného intervalu. Můžeme tedy nastavit zasílání CSNP v jiném intervalu na každém rozhraní.

Pro vygenerování a odeslání CSNP musí být IS na daném rozhraní DIS v případě broadcast rozhraní. Pro PTP linky probíhá odeslání CSNP pouze jednou při úspěšném navázání sousedství.

4.7.2 PSNP

U PSNP je tomu naopak. Pokud je IS na rozhraní DIS, žádná zpráva se negeneruje. V opačném případě projde frontu SSN příznaků a odesílá hlavičku určených LSP.

4.8 SPF

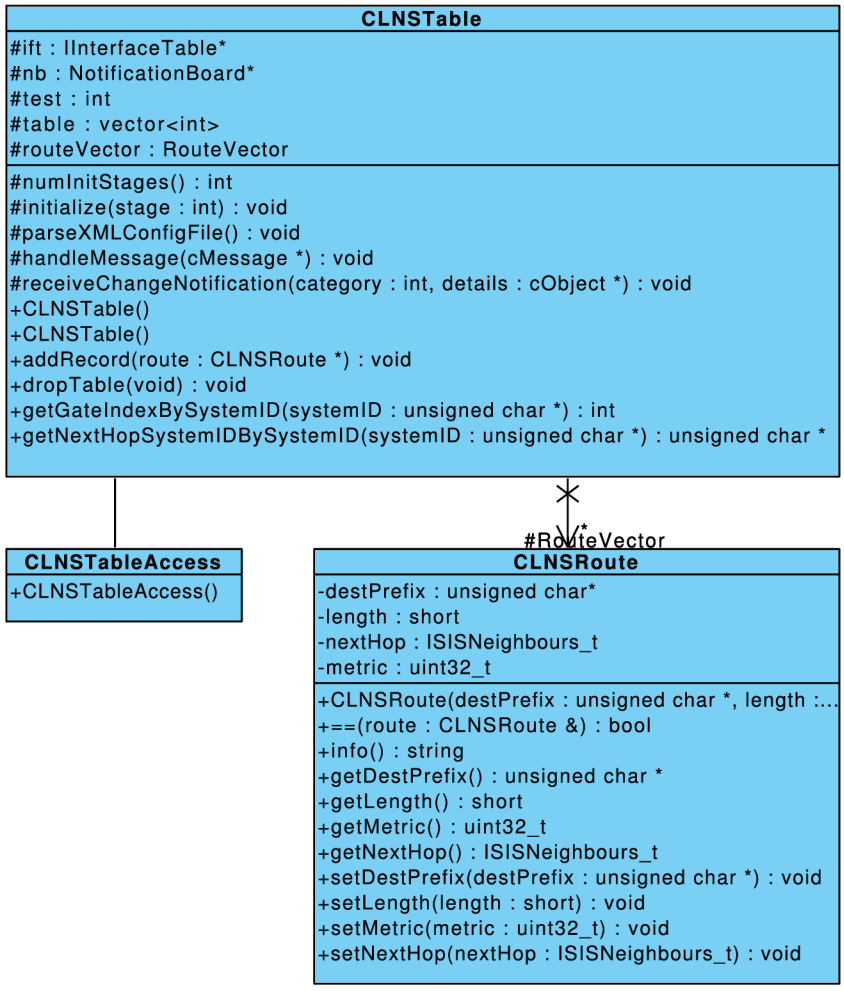
Algoritmus *Shortest Path First* je oproti ISO specifikaci rozšířen o seznam `init`, do kterého vyextrahujeme všechna spojení z LSP databáze. Po vyextrahování celé LSP databáze dojde k ověření, že jsou všechna spojení obousměrná tak, že v seznamu najdeme jak spojení `A->B`, tak i `B->A`. Všechny záznamy, které touto kontrolou neprojdou, jsou ze seznamu vyřazeny. Ve specifikaci je stanoveno, že by tato kontrola měla probíhat při přesunu záznamu položek do seznamu `TENT`. Ale protože po přesunu několika záznamů z `init` bude tento seznam neúplný, ověření by neprobíhalo správně. Další kroky už respektují postup dle specifikace. Aktuálně je implementována podpora pouze pro výpočet nejkratších cest na základě TLV #2 - IS Neighbours. Výpočet je ovšem uzpůsoben pro jednoduché začlenění RFC 1195 a použití TLV #128. Velikost zdroje a cíle je stanovena na 8 B, ačkoliv pro *LAN-ID* stačilo 7 B. Je to

z důvodu, že IP adresa s maskou zabírají právě 8 B. Bude tedy stačit pouze naplnit seznam `init` na základě informací v TLV #128 a ostatní části výpočtu mohou zůstat stejné.

Protože simulace v OMNeTu je diskrétní, nemusíme se zabývat *zmrazením* LSP databáze. V průběhu výpočtu nejkratších cest nemůže dojít ke změně LSP databáze, protože v danou chvíli běží pouze právě výpočet SPF.

4.9 CLNS Table

Vytvořili jsem směrovací tabulku pro CLNS podle již existujících `RoutigTable` pro IPv4 a `RoutingTable6` pro IPv6. Protože žádný jiný modul OSI adresování dosud nepoužíval a nedá se očekávat, že by k tomu došlo, je výsledná implementace zjednodušena. Implementované metody jsou přesně na míru protokolu IS-IS bez větší míry abstrakce. Na ukládání jednotlivých záznamů se využívá třída `CLNSRoute`. Vztah obou tříd je zobrazen na diagramu tříd viz obrázek 4.3. `CLNSTable` je řešena jako samostatný modul dědící z třídy `SimpleModule`.



Obrázek 4.3: Class diagram modulu CLNS

Při pokusu o přidání záznamu do již existujícího cíle se porovná metrika. Původní záznam se přepíše, pokud má nový záznam stejnou, nebo lepší metriku.

`CLNSRoute` umožňuje pro každou destinaci uchovávat několik *next-hop* adres včetně výstupního rozhraní.

Ukázku `CLNSTable` můžete vidět na obrázku 4.4. Obsahuje šest záznamů pro IS se *System-ID* `0100.0000.0005`. Záznamy jsou seřazeny vzestupně podle metriky. Pro každý záznam je uvedeno `InterfaceId` a *System-ID next-hop* IS. Záznam `[3]` a `[5]` mají uvedené dvě *next-hop* adresy.

```
routeVector (std::vector<CLNSRoute *>)
  routeVector[6] (CLNSRoute *)
    [0] = 0100.0000.0005.00/7 -->  metric: 0 via: 0100.0000.0005.00 if=-1 \
    [1] = 0100.0000.0003.00/7 -->  metric: 10 via: 0100.0000.0003.00 if=102 \
    [2] = 0100.0000.0004.00/7 -->  metric: 10 via: 0100.0000.0004.00 if=101 \
    [3] = 0100.0000.0002.00/7 -->  metric: 20 via: 0100.0000.0003.00 if=102 \ 0100.0000.0004.00 if=101 \
    [4] = 0100.0000.0006.00/7 -->  metric: 20 via: 0100.0000.0004.00 if=101 \
    [5] = 0100.0000.0001.00/7 -->  metric: 30 via: 0100.0000.0004.00 if=101 \ 0100.0000.0003.00 if=102 \
```

Obrázek 4.4: Ukázka obsahu CLNS tabulky pro *Level 1* IS.

Obrázek 4.5 zobrazuje CLNS tabulku pro *Level 1-2* IS. Mimo dostupné *Level 1* a *Level 2* IS, obsahuje také záznam pro dostupné *Level 1* oblasti. Prozatím nejsou adresy oblastí odlišeny od adres IS, a v určitých případech by mohlo dojít ke konfliktu. Druhým řešením je použít speciální tabulku pro *Level 2* adresy (*Area-ID*).

```
routeVector (std::vector<CLNSRoute *>)
  routeVector[7] (CLNSRoute *)
    [0] = 4900.0200.0000.00/7 -->  metric: 0 via: 0100.0000.0004.00 if=-1 \
    [1] = 4900.0100.0000.00/7 -->  metric: 10 via: 0100.0000.0002.00 if=101 \
    [2] = 4900.0300.0000.00/7 -->  metric: 10 via: 0100.0000.0005.00 if=102 \
    [3] = 0100.0000.0004.00/7 -->  metric: 0 via: 0100.0000.0004.00 if=-1 \
    [4] = 0100.0000.0002.00/7 -->  metric: 10 via: 0100.0000.0002.00 if=101 \
    [5] = 0100.0000.0005.00/7 -->  metric: 10 via: 0100.0000.0005.00 if=102 \
    [6] = 0100.0000.0006.00/7 -->  metric: 10 via: 0100.0000.0006.00 if=103 \
```

Obrázek 4.5: Ukázka obsahu CLNS tabulky pro *Level 1-2* IS.

4.10 TRILL rozšíření

Výše popsaná implementace protokolu IS-IS modeluje tento protokol podle původní specifikace ISO 10589:2002 pro použití jako L3 směrovacího protokolu. V této části si popíšeme rozšíření pro podporu protokolu TRILL.

Je zaveden nový parametr `mode`, podle kterého se určuje, zda protokol běží jako L2, nebo L3. Nastavení se provádí na základě `deviceId`, které musí mít každé zařízení a je specifikováno v NED souboru. V případě, že se jedná o `Router`, nastaví se `L3_ISIS_MODE`. Pro `RBridge` se zvolí `L2_ISIS_MODE`.

Byl vytvořen nový typ Hello zpráv `TRILLHello` a upraven stavový automat pro navazování sousedství.

V rámci rozšíření pro TRILL byla zavedena nová třída `ISISInterfaceData`, která sdružuje všechny parametry IS-IS, které jsou konfigurovatelné *per interface*. Dědí z třídy `InterfaceProtocolData` a přes `InterfaceEntry` se váže přímo na `InterfaceTable`. Do `ISISInterfaceData` jsou postupně převáděny i všechny parametry z `ISISinterface`.

V L2 režimu není povinná žádná konfigurace, dokonce ani NET adresa. Pokud není zadaná, RBridge si ji vygeneruje. *System-ID* nastaví na hodnotu nenulové MAC adresy, kterou nalezne, při procházení tabulky rozhraní. Jako *Area-ID* zvolí pevně danou adresu 0 a stejně tak i NSEL.

4.11 TRILL Hello

Generování a rozesílání TRILL Hello zpráv je řešeno odlišně oproti LAN a PTP Hello. Protože TRILL Hello jsou komplikovanější, byla jejím generováním věnována větší pozornost. Všechna rozhraní mají svoje TRILL Hello zprávy. Aby nemusely být generovány při každém odesílání, vytvořené Hello zprávy (pro každou povolenou VLAN jedna viz kód 3.1) jsou uloženy ve třídě `ISISInterfaceData`.

Ke generování dochází pouze pokud je některý parametr změněn. V takovém případě se zneplatní příznak `helloValid`. Při vypršení `helloInterval` je nejprve ověřen `helloValid`. Pokud je platný, dojde pouze k vytvoření kopie a odeslání na dané rozhraní. V opačném případě jsou předchozí Hello smazány a vygenerovány nové. Generování neprobíhá okamžitě po zneplatnění `helloValid` příznaku, ale až při samotném rozesílání. Může tak proběhnout několik změn, ale pouze jedno generování. V nejlepším případě dojde ke generování pouze několikrát na začátku simulace než dojde ke zkonvergování, a poté už se jen rozesílají předem vygenerované hello zprávy. V nejhorším případě bude docházet ke generování stejně často jako pro LAN a PTP Hello.

4.12 DRB a Appointed Forwarder

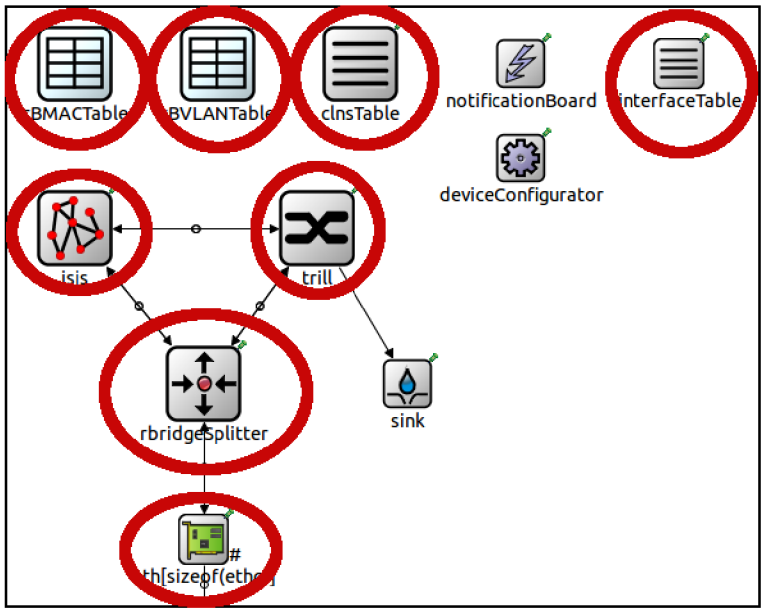
Volba DRB probíhá stejně jako volba DIS, ale vítěz se navíc nastaví i jako *Appointed Forwarder* pro povolené VLAN. Seznam povolených VLAN najde ve třídě `TRILLInterfaceData`.

4.13 Generování TRILL LSP

Generování TRILL LSP probíhá stejně jako v případě L3 IS-IS. V přenášených LSP používáme TLV #2, které je ale v L2 zakázané a mělo by se používat TLV #22. Jedná se o tzv. *wide-metric* TLV, které nabízí lepší škálovatelnost metriky linky.

4.14 RBridge

Vytvořený model RBridge vychází ze struktury klasického přepínače. Na obrázku 4.6 můžete vidět strukturu RBridge společně se zvýrazněnými moduly, které jsou nově vytvořeny, nebo byly upraveny pro podporu protokolu TRILL. `L3` funkcionalitu zajišťuje modul `ISIS`, pracující v režimu `L2_ISIS_MODE`, tedy aktivovanými TRILL rozšířeními. Základní `L2` funkcionalitu řeší modul `TRILL` ve spolupráci s `RBMACTable` `RBVLANTable`. Diagram tříd z RBridge modulu je uveden v příloze.

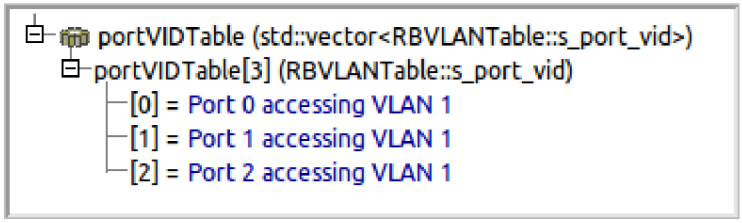


Obrázek 4.6: Model RBridge.

4.14.1 RBVLANTable

`RBVLANTable` je převzatá z modelu přepínače `ANSASwitch`. Uchovává informace o provozovaných VLAN jako jsou jméno a VLAN ID. Je možné vyhledávat jak VLAN ID podle portu, tak porty patřící do určité VLAN.

Na obrázku 4.7 vidíme výpstup `RBVLANTable` pro RBridge se třemi porty patřícími do VLAN ID 1.



Obrázek 4.7: Ukázka výstupu `RBVLANTable`.

4.14.2 RBMACTable

`RBMACTable` je také převzatá z `ANSASwitch`, ale doznala několika změn. Slouží jako směrovací tabulka pro L3 protokoly. Obsahuje množinu výstupních portů, případně specifikuje *nickname* pro *next-hop* RBridge na cestě k cíli, pro dvojice MAC adresa, VLAN ID. Nově `ESTRecord` (**E**nd **S**tation **T**able **R**ecord) obsahuje proměnnou `type`. `type` určuje typ daného záznamu a může mít jednu z následujících hodnot:

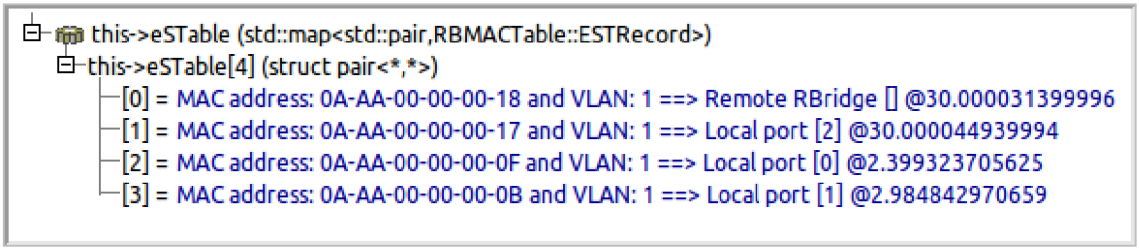
- `EST_EMPTY` – se používá pro označení prázdného záznamu, kdy pro danou kombinaci MAC adresa, VLAN ID, není v tabulce záznam,

- `EST_LOCAL_PROCESS` – značí, že rámec je určen k lokálnímu zpracování na daném RBridge,
- `EST_LOCAL_PORT` – znamená, že cíl je přpojen k některému lokálnímu portu, a rámec se posílá v nativní podobě,
- `EST_RBRIDGE` – adresa byla naučena při rozbalení TRILL – zapouzdřeného rámce, nebo pomocí ESADI protokolu (aktuálně není implementován) a místo množiny výstupních portů se použije `ingressNickname` jako *next-hop* pro TRILL-zapouzdřený rámec a
- `EST_MULTICAST` – pro doručování *multi-destination* rámců specifikuje jak výstupní lokální porty, tak případně i *nickname*.

Na obrázku 4.8 je vidět výstup z `RBMACTable` se čtyřmi záznamy. Spodní dva záznamy, označené jako `[2]` a `[3]` označují dva sousední RBridge připojené na lokálním portu `[0]` a `[1]`. Poslední parametr určuje simulační čas v kterém byla adresa naučena.

Záznam `[1]` patří lokálně připojené stanici.

A poslední položka `[0]` byla naučena z TRILL-zapouzdřeného rámce, což signalizuje `Remote RBridge`, v čase přibližně 30 s.



Obrázek 4.8: Ukázka výstupu `RBMACTable`.

4.14.3 InterfaceTable

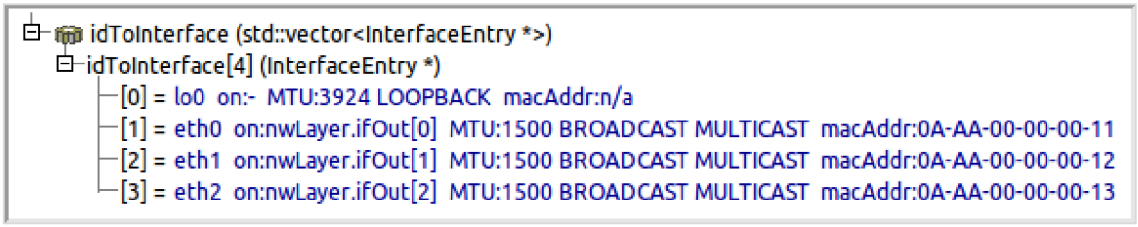
Do položek `InterfaceTable`, která je součástí balíku INET, jsme pouze přidávali odkazy na `TRILLInterfaceData *trillData` a `ISISInterfaceData *isisData`. Oba objekty obsahují různé konfigurovatelné parametry pro daný protokol a rozhraní. Při inicializaci obou objektů se nastavuje kontext na rozhraní, pro které je daný objekt určený. Můžeme tak přistupovat jak k parametrům rozhraní přes `InterfaceTable`, tak i k objektu rozhraní, pro které máme parametry v podobě např. `isisData` objektu.

Jak vypadá výstup `InterfaceTable` můžeme vidět na obrázku 4.9. Jedná se o RBridge se třemi *Ethernet* rozhraními a jedním virtuálním rozhraním *loopback*. *Loopback* rozhraní nemá MAC adresu, ani specifikované rozhraní a je automaticky generováno pro každé zařízení implicitně tabulkou rozhraní.

Příchozí rámec nejprve prochází modulem `RBEthernetInterface`.

4.14.4 RBEthernetInterface

Modul `RBEthernetInterface` vychází ze stávajícího `EthernetInterface` definovaného v rámci balíku INET, ale nezajišťuje zapouzdřování. Tato drobná změna oproti rozhraní použitém v `ANSASwtich` byla nutná z důvodu zavedení `InterfaceTable` do RBridge. Při



Obrázek 4.9: Ukázka výstupu `InterfaceTable`.

registraci rozhraní pro *Medium Access Control* modul se generuje jméno na základě numerického suffixu nadřazeného modulu. V případě `ANSASwitch` je nadřazeným modulem pro `MAC` přímo modul přepínače a docházelo by tak k registraci dvou rozhraní se stejným jménem. Byl zachován modul řešící kontrolu pro přístup k médiu (`EtherMAC`) s výstupní frontou. Odchozí rámec je zařazen do fronty, dokud není médium připraveno na přenos. Příchozí rámec je pouze přeposlán do modulu `RBSplitter`.

4.14.5 RBSplitter

`RBSplitter` rozděluje provoz mezi `TRILL` a `ISIS` a slouží jako místo pro možnost budoucí integrace dalších protokolů. Pro modul `ISIS` zajištuje prozatím i *Ethernet* zapouzdřování. Rozdělení provozu probíhá na základě kontroly *Ethertype*. `ETHERTYPE_L2_ISIS` rámce jsou rozbaleny a zaslány do modulu `ISIS` a všechny ostatní jsou včetně *Ethernet* hlavičky zaslány do `TRILL`.

`RBSplitter` nemá žádné parametry konfigurovatelné pomocí XML souboru.

4.14.6 ISIS

`ISIS` zajišťuje `L3` funkcionalitu od Hello zpráv až po počítání SPF a naplnění CLNS tabulky. Pracuje v `L2_ISIS_MODE` režimu.

Jeho činnost je popsána na začátku této kapitoly a doplněna popisem TRILL rozšíření.

4.14.7 TRILL

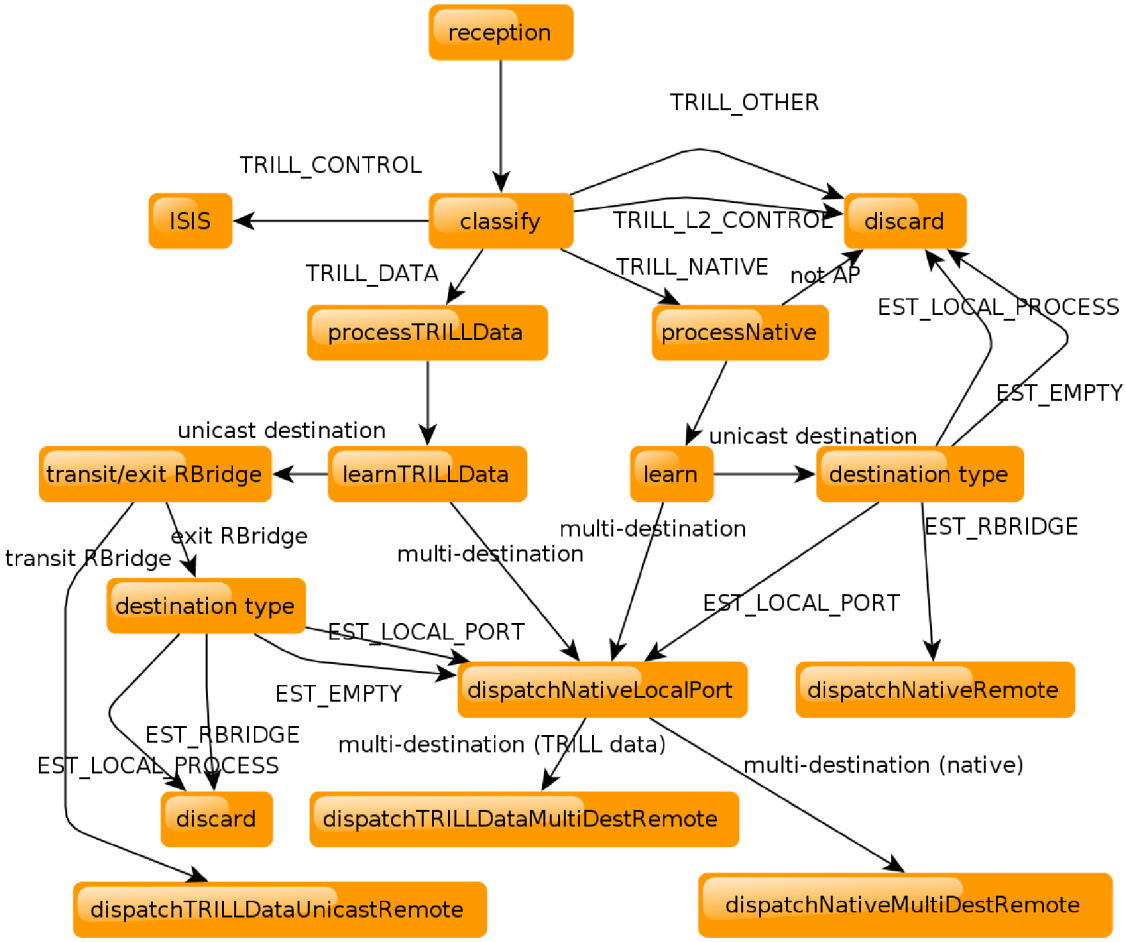
Činnost modulu `TRILL` je rozdělena na 4 základní části.

Přijetí (`reception`), klasifikování (`classify`), zpracování (`process`) a odeslání (`dispatch`) rámce. Zpracování a odeslání probíhá odlišně podle typu z klasifikace a typu destinace ve vztahu ke zpracovávajícímu RBridge.

Celkový průběh zpracování příchozího rámce modulem `TRILL` můžete vidět na obrázku 4.10.

Přijetí

Přijetí spočívá v naplnění struktury `tFrameDescriptor`, reprezentující příchozí rámec. Pokud se jedná o `AnsaEthernetFrame`, nastaví se VLAN tag obsažený přímo v rámci. Pro `EthernetIIFrame` se použije VLAN ID z `RBVLANTable` podle portu, na kterém byl rámec přijat.



Obrázek 4.10: Work-flow diagram zpracování příchozího rámce modulem `TRILL`.

Klasifikování

Klasifikování jej označí jako jeden z pěti typů zmíněných v kapitole 3 a zavolá se odpovídající metoda na zpracování. Zpracování probíhá na základě klasifikovaného typu takto:

- `TRILL_L2_CONTROL` – není implementováno a rámec se zahazuje,
- `TRILL_DATA` je zpracován metodou `processTRILLData`,
- `TRILL_NATIVE` zpracovává `processNative`,
- `TRILL_OTHER` se zahazuje v souladu se specifikací a
- `TRILL_CONTROL` zpracováná modul `ISIS`.

Mimo zmíněné typy je zaveden ještě `TRILL_NONE` pro detekci případné chybné klasifikace. Zpracování `TRILL_L2_CONTROL` rámců v čistě RBridge campusu není potřeba. Jedná se o zprávy protokolů jako STP, které ovlivňují chování protokolu TRILL ve smíšené topologii při migraci z klasických přepínačů.

Zpracování

TRILL modul řeší zpracování pouze TRILL_NATIVE a TRILL_DATA rámců. Ostatní se buď zahazují, protože jejich zpracování není implementováno (TRILL_L2_CONTROL), nebo protože to tak definuje specifikace (TRILL_OTHER), nebo je řeší jiný modul (TRILL_CONTROL).

Pro oba typy je nejprve provedena kontrola zdrojové adresy. Pokud se jedná o unicastovou adresu, zavolá se odpovídající metoda learn, pro vytvoření, nebo aktualizaci záznamu v RBMACTable.

Aby mohl být nativní rámec zpracován, musí být RBridge na příchozím portu *Appointed Forwarder* pro danou VLAN. Následně je nativní provoz rozdělen podle typu záznamu pro dvojici cílová MAC adresa, VLAN ID z RBMACTable.

V rámci zpracování se provede nejprve zjištění možných výstupních portů. Podle zjištěného typu destinace, a podle odpovídající kombinace vstupního a výstupního zapouzdření, se množina výstupních portů zredukuje. Jsou možné celkem čtyři možnosti zapouzdření:

1. NATIVE → NATIVE
2. NATIVE → TRILL_DATA
3. TRILL_DATA → TRILL_DATA
4. TRILL_DATA → NATIVE

Metody provádějící snížení počtů výstupních portů začínají klíčovým slovem egress. Konkrétní metoda záleží také na tom, zda se jedná o *unicast*, nebo o *multi-destination* provoz.

V případě *multi-destination* provozu se nezjišťuje typ cíle, protože se stejně rámec duplikuje na všechna rozhraní, která splňují výstupní požadavky. Pro nativní provoz je to například *Appointed Forwarder* na rozhraní atd.

U TRILL_DATA rámců záleží, jestli jsou pouze přeposílány, nebo určené pro zpracovávající RBridge. Přeposílaným rámcům je snížen hopCount a předány k odeslání. TRILL zapouzdřený provoz je roztříděn podle typu destinace. EST_LOCAL_PROCESS a EST_EMPTY jsou zahozeny a EST_LOCAL_PORT je předán k odeslání. Typ EST_EMPTY je vrácen v případě, že došlo k přesunu koncové stanice, nebo k vyexpirování záznamu z RBMACTable. V takovém případě je rámec předán k odeslání jako *multi-destination*, ale pouze v nativní formě.

Podle kombinace vstupního-výstupního zapouzdření a typu provozu je použita odpovídající metoda na odeslání.

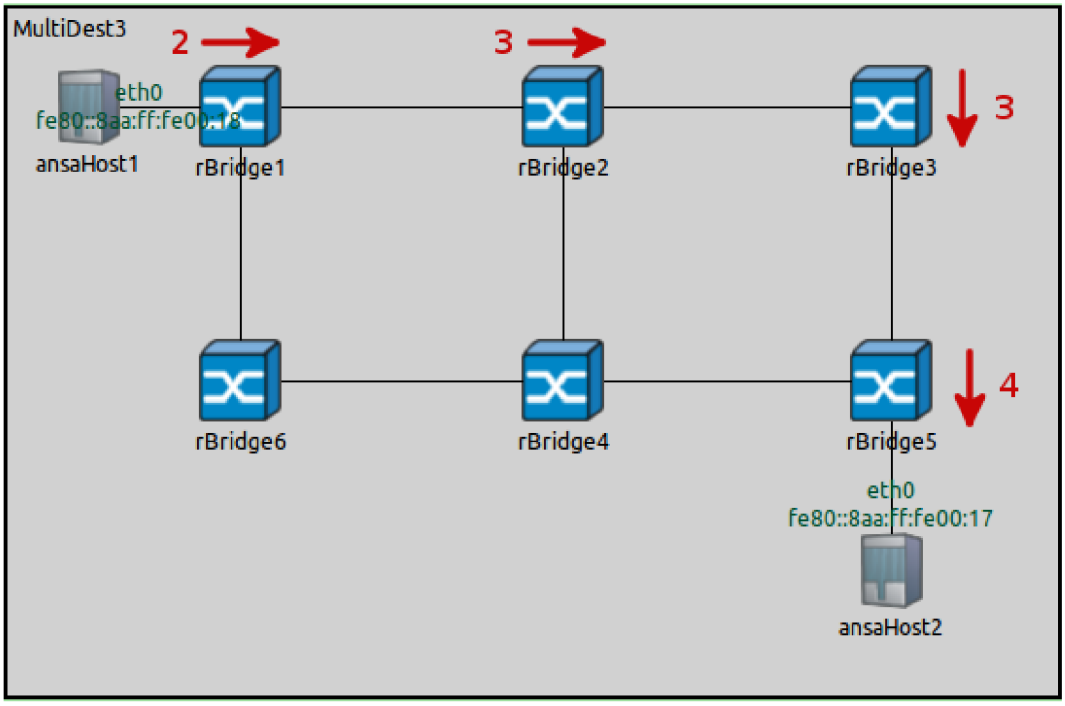
Na obrázku 4.11 je zvýrazněno použité zapouzdřování pro unicastovou komunikaci od AnsaHost1 vlevo nahoře do AnsaHost2 vpravo dole.

Odeslání

Metody zajištující odeslání začínají na dispatch. Nativním rámcům přidávají *Ethernet* hlavičku, TRILL rámcům navíc ještě TRILL hlavičku a vnější *Ethernet* hlavičku. Konkrétní metoda odeslání rámce závisí na kombinaci použitého zapouzdření vstupního a výstupního rámce. Výstupní zapouzdření určuje typ destinace vzhledem k RBridge zpracovávající daný rámec.

Nativní rámec s *multi-destination* cílovou adresou se nejprve odešle na všechny lokální porty, kde je *Appointed Forwarder*, pomocí dispatchNativeLocalPort a na všechny porty, kde má navázáno sousedství pomocí dispatchNativeMultiDestRemote.

TRILL *multi-destination* rámce se nejprve odešlou do dispatchNativeLocalPort a poté do dispatchTRILLDataMultiDestRemote.



Obrázek 4.11: Jednoduchá topologie se zvýrazněným použitým zapouzdřením.

4.15 Možná rozšíření

V této části popíšeme způsoby jakými je možno vylepšit stávající implementaci ať už z hlediska lepší korespondence se specifikací, nebo z hlediska zavedení nové funkcionality.

4.15.1 Generování LSP

Toto rozšíření má za úkol eliminovat problém s generováním LSP představeným v sekci 4.6.1.

Každý objekt, který se vkládá do LSP, by byl opatřen referencí na LSP, kterým jsou jeho data přenášena a příznakem udávajícím změnu vkládaného objektu.

Získáme tím vazbu na to, v kterém LSP se nacházela daná informace při předchozím generování a budeme vědět, které informace se změnily. Pokud budou mít všechny objekty nastavené, že nedošlo ke změně, bude stačit u všech vlastních LSP v databázi nastavit *Remaining Lifetime* a inkrementovat *Sequence Number*. Pokud budou některé objekty změněné a jejich velikost se zvětší natolik, že by se nevešly do původního LSP, vytvoří se nové LSP na konci aktuálního rozsahu LSP.

4.15.2 Autentizace a kontrolní součet

Pokud bychom chtěli provádět útoky, nebo testovat zpracování poškozených zpráv, hodila by se autentizace a kontrolní součet.

Kontrolní součet je povinná součást protokolu IS-IS, ale pro běžné provozování v simulátoru, který zaručuje spolehlivé doručování, není kritické. Samotné pole je v patřičných zprávách připravené a v kódu jsou označena místa, kde se má kontrola provádět.

Autentizace sice povinná není, ale její použití v simulaci by mohlo být obzvlášť žádané. Autentizace je řešena pomocí TLV a místa na kterých by se TLV #10 *Authentication* mělo přidávat jsou označena pomocí `TODO` štítků.

4.15.3 Podpora IPv4

Nejvyužitelnějším rozšířením by jistě bylo zavedení podpory IP dle RFC 1195 [4] přesněji zavedení podpory TCP/IP pro duální prostředí. IP rozšíření dle RFC 1195.

4.15.4 MTU test

Aktuální implementace nepodporuje testování MTU ať už pomocí zvětšování Hello zpráv L3, nebo pomocí nových zpráv v L2.

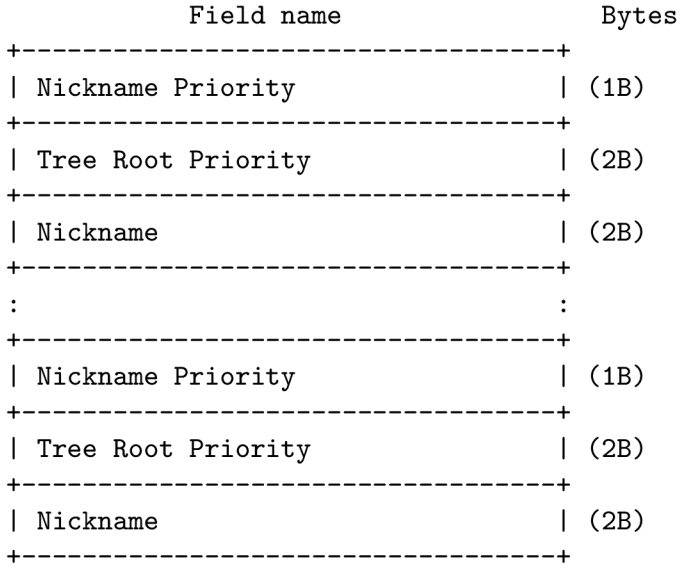
Obě varianty jsou relativně snadno implementovatelné. V případě L3 varianty by stačilo vytvořit kód pro přidávání *Padding* TLV do Hello zpráv.

Pro způsob používaný v L2 bychom museli vytvořit nový typ časovačů, které by iniciovaly MTU test a metody pro vytváření a zpracování MTU Request zpráv.

4.15.5 Výměna Nickname

Jedná se o rozšíření pro podporu protokolu TRILL. Dynamická výměna *přezdívek* zajišťuje distribuci všech přezdívek napříč celou topologií pomocí TLV #242 Router CAPABILITY, respektive sub-TLV #6 *Nickname*.

Nickname sub-TLV #242.6 Jeho struktura je uvedena na obrázku 4.12. Obsahuje seznam používaných *přezdívek* společně



Obrázek 4.12: Formát sub-TLV #242.6 *Nickname*

- **Nickname Priority** je 8 b hodnota udávající prioritu pro držení níže uvedeného *Nickname*.

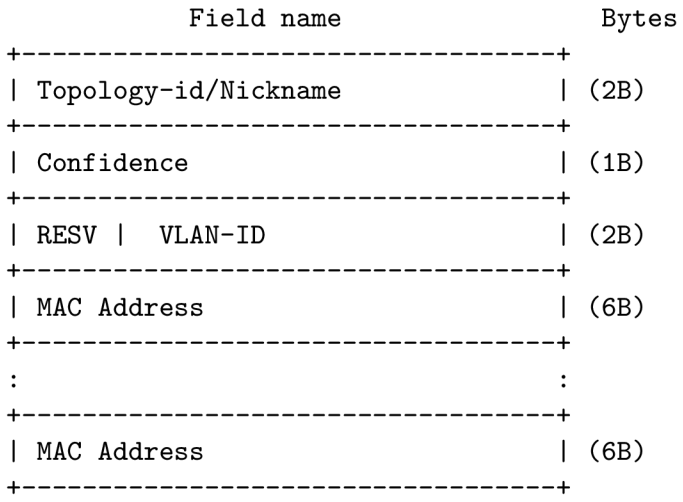
- **Tree Root Priority** určuje prioritu k tomu, aby se toto *Nickname* použilo jako kořen distribučního stromu pro doručování *multi-destination* provozu.
- **Nickname** je *přezdívka*, ke které se vztahují výše uvedené parametry.

4.15.6 ESADI

Zavedení protokolu ESADI by vyžadovalo drobné změny jak při vytváření sousedství, tak při práci s LSP. Zahrnovalo by to zavedení parametru, podle kterého bychom poznali, že se jedná ESADI zprávy. Museli bychom vytvořit další tabulku sousedství a LSP databázi. Z ní by se zjištěné informace propagovaly do stávající `RBMACTable`.

Do generování by stačilo pouze doplnit kód pro TLV #147. Odesílání všech ESADI zpráv by pak muselo chodit přes modul TRILL, který by jim přidal TRILL hlavičku.

Struktura TLV #147 je uvedena na obrázku 4.13. Obsahuje seznam používaných *přezdívek* společně



Obrázek 4.13: Formát sub-TLV #147 *MAC-Reachability*

- **Topology-id/Nickname** má význam závislý na technologii v rámci které je toto TLV používáno. Pokud je toto pole nastaveno na nulu, znamená to, že uvedené MAC adresy jsou dostupné pro všechny topologie/*přezdívky*.
- **Confidence** udává míru důvěry v přenášené MAC adresy.
- **VLAN-ID** identifikuje VLAN, do které patří všechny uvedené MAC adresy.
- **MAC Address** je seznam dostupných MAC adres pro uvedenou topologii, nebo *nickname*.

Kapitola 5

Testování

V této kapitole si na jednoduchých příkladech a simulacích ukážeme funkčnost modelovaných protokolů.

5.1 IS-IS

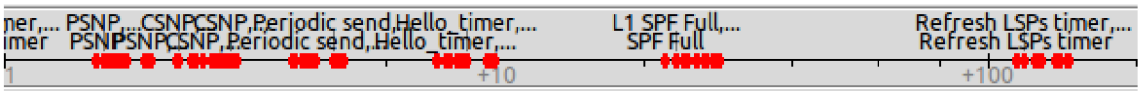
Pokud není explicitně uvedeno jinak, mají všechny linky výchozí metriku 10.

5.1.1 Rozprostření zpráv v čase

V této části si ukážeme vliv aplikování *jitter* při plánování zpráv zmíněné v sekci 2.15.2. Jedná se o časovou osu s logaritmickou stupnicí. Červené tečky představují naplánovanou zprávu. Výřez časové osy zobrazuje naplánované zprávy v časovém rozmezí 1 - 100 s od aktuálního času v simulaci.

Na obrázcích 5.1 a 5.3 je zobrazeno použití bez *jitter* v simulačním čase 1 s a 165 s. Jde vidět, že většina zpráv je soustředěna do jednotlivých časových okamžiků.

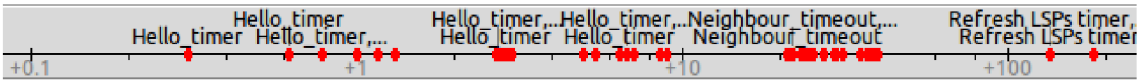
Obrázek 5.1: Rozprostření zpráv bez *jitter* v čase 1 s.



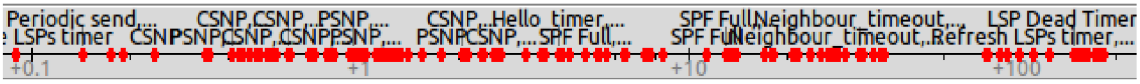
Obrázek 5.2: Rozprostření zpráv s aktivovaným *jitter* v čase 1 s.

Na druhé straně u obrázků 5.2 a 5.4 vidíme stav plánovaní zpráv s použitím *jitter* opět v časech 1 s a 165 s. S postupným časem se rozprostření zpráv ještě více zvýrazňuje.

Implementování jitter ma za výhodu přiblížení se ke specifikaci, která *jitter* výslovně vyžaduje. Výhoda samotného *jitter* spočívá v rozložení zpráv a např při vypršení *životnosti* u LSP nedojde k vygenerování *purge LSP* všemi IS v topologii.



Obrázek 5.3: Rozprostření zpráv bez *jitter* v čase 165 s.

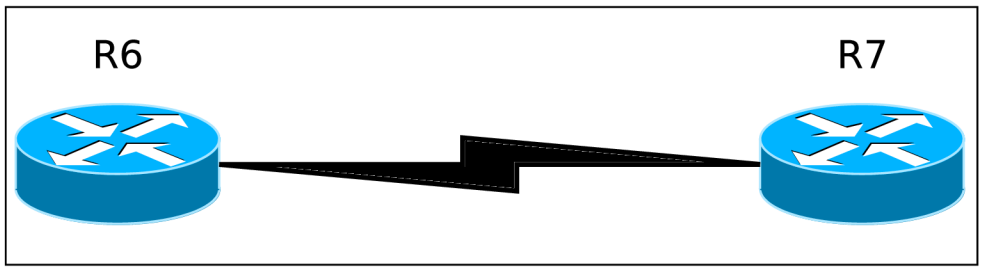


Obrázek 5.4: Rozprostření zpráv s aktivovaným *jitter* v čase 165 s.

5.2 ISIS na Point-to-point

5.2.1 Topologie

V topologii, viz obrázek 5.5, se nachází dva IS (směrovače) `R6` a `R7`.



Obrázek 5.5: Point-to-point topologie

V OMNeT simulaci využíváme přímé propojení pomocí *Ethernet* emulující *point-to-point* spojení. V reálné topologii byly využity dva směrovače Cisco 2691 a propojení bylo realizováno sériovou linkou s výchozím zapouzdřením HDLC. Použitá zapouzdření nemají vliv na činnost simulace z hlediska pozorovaných faktorů.

Směrovač `R6` má nastavený NET na `49.0002.0100.0000.0006.00` a `R7` na `49.0002.0100.0000.0006.00`. Oba pracují pouze na *Level 1* a patří do oblasti `49.0002`.

5.2.2 Scénář

Oba IS si začnou navzájem zasílat Hello zprávy a budeme pozorovat jakým způsobem probíhá navazování sousedství. Po úspěšném navázání sousedství dojde k synchronizaci LSP databází.

5.2.3 Reálná síť

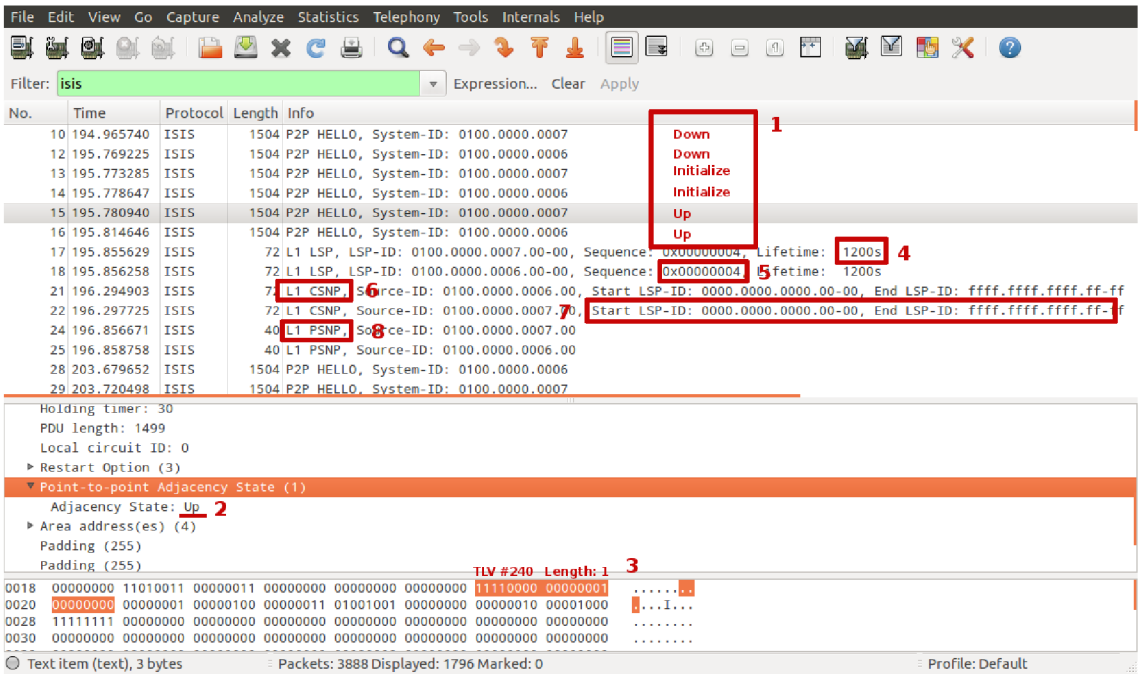
Na obou směrovačích byl zprovozněn protokol IS-IS pomocí následující sady příkazů v kódu 5.1.

Na obrázku 5.6 je zachycena komunikace mezi zmíněnými IS.

Cisco implementace protokolu IS-IS automaticky provozuje i ES-IS [12], který v našem řešení zahrnut není, protože neplánujeme provozovat simulace OSI provozu, ale pouze využití jako směrovacího protokolu pro směrování dalších (IPv4, IPv6) síťových protokolů.

Kód 5.1: Sada příkazů pro zprovoznění IS-IS

```
R6>en
R6#
R6(config)#router isis
R6(config-router)#net 49.0002.0100.0000.0006.00
R6(config-router)#is-type Level-1
R6(config-router)#interface serial 0/0
R6(config-if)#clns router isis
R6(config-if)#no shut
```



Obrázek 5.6: Záznam komunikace mezi směrovači `R6` a `R7`.

ES-IS je pro OSI CLNP, to, co DHCP pro IPv4 a DHCPv6 s NDP pro IPv6. Zprávy protokolu ES-IS byly ze zobrazeného výstupu odfiltrovány.

Pro lepší orientaci jsou na obrázku zvýrazněny některé části.

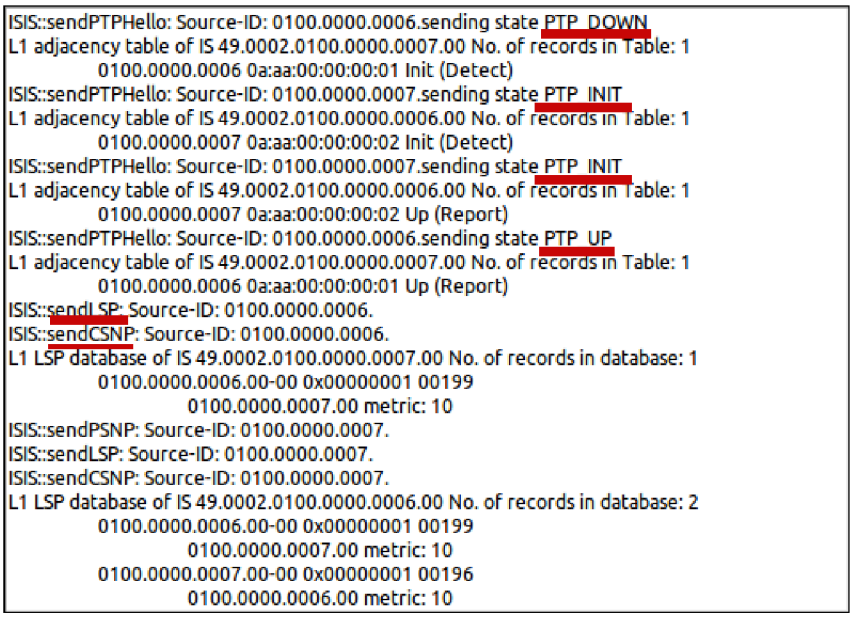
1. První obdélník ukazuje stav sousedství zasílaný v dané zprávě.
2. Druhý zobrazuje stav `Up` v jedné ze zpráv.
3. V třetí části je zvýrazněno TLV #240.
4. Po úspěšném navázaní sousedství dochází k zaslání vygenerovaných LSP s výchozí *životností* 1200s.
5. Protože před samotným povolením rozhraní pomocí `no shutdown`, muselo proběhnout několik dalších příkazů, není sekvenční číslo 1, ale začíná na hodnotě 4.
6. Zaslání LSP iniciuje prvotní synchronizaci databáze pomocí CSNP zpráv.
7. Protože se všechny LSP záznamy vejdou do jedné zprávy, je `Start LSP-ID` a `End LSP-ID` nastaveno na plný rozsah.
8. A následně dojde k potvrzení přijatých LSP pomocí CSNP zprávy.

Obrázek 5.7 znázorňuje výpis LSP databáze na směrovači `R7` pomocí příkazu `do show isis data` z globálního konfiguračního režimu. LSP databáze obsahuje nejenom záznamy pro IS, ale také pro ES adresu a hostname.

```
R7(config)#do show isis data

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R6.00-00              0x0000005C   0xB547        1174              0/0/0
R7.00-00            * 0x0000005A   0xC933        1195              0/0/0
R7(config)#do show isis data det

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R6.00-00              0x0000005C   0xB547        1168              0/0/0
  Area Address: 49.0002
  Hostname: R6
  Metric: 10         IS R7.00
  Metric: 0          ES R6
R7.00-00            * 0x0000005A   0xC933        1189              0/0/0
  Area Address: 49.0002
  Hostname: R7
  Metric: 10         IS R6.00
  Metric: 0          ES R7
R7(config)#
```

Obrázek 5.7: Výpis LSP databáze na směrovači `R7`.

Na obrázku 5.8 vidíme výstup ze simulace v OMNeTu. Z výstupu byly odfiltrovány zprávy ostatních modulů. U zasílaných PTP Hello zpráv je zvýrazněn stav sousedství, který se posílá v TLV #240. Obsah `Adjacency table` se vypisuje vždy při přijetí Hello PDU. Poté, co sousedství přejde do stavu `Up` (`Report`). Odešle `R6` svoje LSP, což iniciuje zaslání i CSNP zprávy. `R7` po přijetí a zpracování LSP, vypíše stav svojí LSP databáze a odešle potvrzení přijetí LSP ve formě PSNP. Stejná činnost následuje i u druhého směrovače.



Obrázek 5.8: Výstup modulu IS-IS v OMNeTu pro point-to-point spojení.

5.2.4 Srovnání

Oproti reálné topologii se poslalo o jedno PTP Hello méně, aby došlo k vytvoření sousedství ve stavu `Up`. To je způsobeno tím, že oba směrovače zašlou svoje první Hello takřka okamžitě a protože o sobě ještě nevědí, jsou ve stavu `Down`. Naopak v simulaci stačí dorazit první Hello zpráva dříve než druhý směrovač stihne vygenerovat svoji Hello zpráv. V dalším postupu už se shodují.

U rozesílání LSP je v simulaci pořadí zpráv částečně jiné. Na rozdíl od reálné topologie, předběhla PSNP zpráva potvrzující přijetí LSP, odeslání vlastního LSP. Pořadí zpráv z hlediska zasílání LSP a jejich potvrzení je chování stejné.

Obsah LSP databáze z reálné topologie obsahuje navíc ES záznamy, díky provozování ES-IS.

5.3 IS-IS na LAN

V tomto příkladu si ukážeme provozování IS-IS na LAN segmentu.

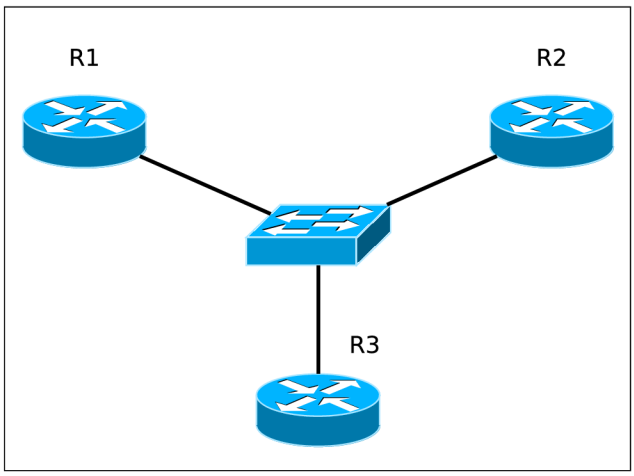
5.3.1 Topologie

V této topologii jsou tři směrovače propojené pomocí jednoho přepínače viz obrázek 5.9. Směrovače jsou součástí jedné *L1* oblasti s adresou `49.0001`.

Relevantní parametry IS z LAN topologie jsou v tabulce 5.1.

5.3.2 Scénář

Všichni tři IS začnou odesílat LAN Hello zprávy s nastaveným DIS na svoje *LAN-ID*. Po vytvoření sousedství dojde k volbě DIS. Rozhodnutí proběhne na základě porovnání priority



Obrázek 5.9: Point-to-point topologie.

Model	Hostname	NET	Level	Priority
Cisco 2691	R1	49.0001.0100.0000.0001.00	1	100
Cisco 2691	R2	49.0001.0100.0000.0002.00	1	100
Cisco 2691	R3	49.0001.0100.0000.0003.00	1	64

Tabulka 5.1: Konfigurace ISs v ukázce provozovaní IS-IS na LAN.

i MAC adresy. Pomocí CSNP a PSNP zpráv dojde k synchronizaci LSP databází a nakonec vytvoření nejkratších cest.

5.3.3 Reálná síť

No.	Time	Protocol	Length	Info
2	0.326	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0001
4	0.497	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0002
8	1.106	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0003
9	1.326	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0001
12	1.495	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0002
13	1.578	ISIS	80	L1 LSP, LSP-ID: 0100.0000.0001.00-00, Sequence: 0x00
17	2.131	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0003
18	2.357	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0001
20	2.512	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0002
21	2.622	ISIS	80	L1 LSP, LSP-ID: 0100.0000.0003.00-00, Sequence: 0x00
23	3.137	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0003
24	3.174	ISIS	86	L1 LSP, LSP-ID: 0100.0000.0002.01-00, Sequence: 0x00
25	5.208	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0002
27	8.415	ISIS	1514	L1 HELLO, System-ID: 0100.0000.0002

Obrázek 5.10: Záznam komunikace mezi přepínačem a IS `R2` v LAN topologii.

Obrázek 5.10 zachycuje komunikaci na lince mezi přepínačem a směrovačem `R2` z topo-

Kód 5.2: Obsah LSP databáze na směrovači R2.

```
R2#show isis database details
IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00              0x00000002   0x3B30        1040              0/0/0
  Area Address: 49.0001
  Hostname: R1
  Metric: 10          IS R2.01
  Metric: 0           ES R1
R2.00-00            * 0x00000002   0x6305        1041              0/0/0
  Area Address: 49.0001
  Hostname: R2
  Metric: 10          IS R2.01
  Metric: 0           ES R2
R2.01-00            * 0x00000002   0xAA14        1041              0/0/0
  Metric: 0           IS R2.00
  Metric: 0           IS R3.00
  Metric: 0           IS R1.00
R3.00-00              0x00000002   0x8BD9        1038              0/0/0
  Area Address: 49.0001
  Hostname: R3
  Metric: 10          IS R2.01
  Metric: 0           ES R3
```

Kód 5.3: Výpis sousedů na směrovači R2.

```
R2#show isis neighbors
System Id      Type Interface   IP Address      State Holdtime Circuit Id
R3             L1   Fa0/0                       UP    26       R2.01
R1             L1   Fa0/0                       UP    27       R2.01
```

logie na obrázku 5.9.

Postupně si popíšeme jednotlivé zvýrazněné části.

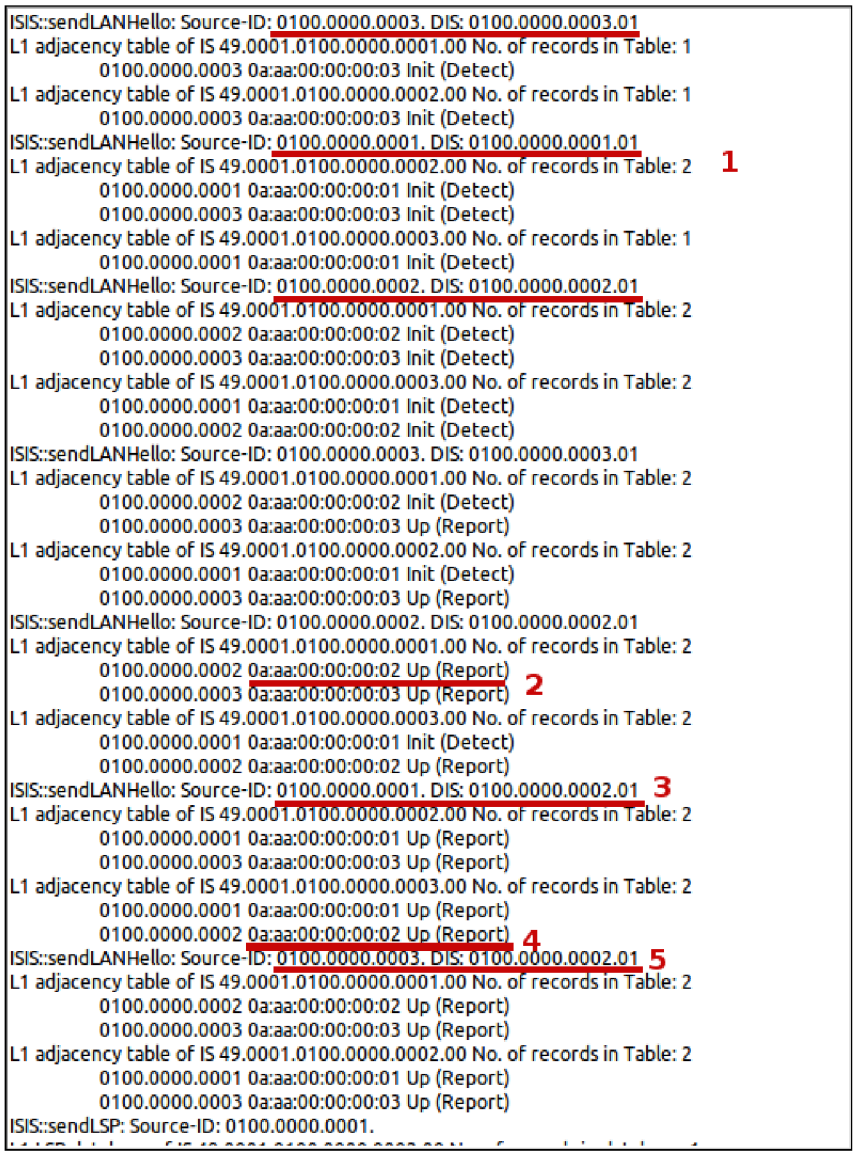
1. Všechny IS rozesílají Hello PDU se svým *LAN-ID* jako DIS.
2. Po navázání sousedství dochází na `R1` k vygenerování a odeslání LSP.
3. V následující Hello zprávě posílá `R1` adresu `R2` jako DIS.
4. K vytvoření sousedství dochází i na `R3`.
5. `R3` si aktualizuje DIS a posílá je ve svém Hello PDU.

V kódu 5.2 je výpime jednotlivé s sousedů na směrovači `R2` po zkonvergování topologie. Celá topologie zkonvergovala v čase 10.27 s.

V kódu 5.3 jsou uvedeny sousední IS pro směrovač `R2`.

5.3.4 Simulace

Na obrázku 5.11 vidíme výpis ze simulace v OMNeTu pro topologii z obrázku 5.9. Ve výpisu jsou uvedeny pouze relevantní výpisy modulu `ISIS`.



Obrázek 5.11: Navazování sousedství a volba DIS na LAN.

Nyní si popíšeme jednotlivé zvýrazněné části.

1. Tři podtržené řádky vyjadřují odeslání LAN Hello PDU pro IS s uvedeným *Source-ID* a adresou pro DIS. Nejprve všechny IS zasílají svoji adresu jako DIS.

2. Dokud nemají IS `R1` a `R3` navázané sousedství s `R2` ve stavu `Report`, ignorují jeho vyšší prioritu. Po úspěšném vytvoření sousedství, již `R1` pracuje se správným DIS.

3. Ve svých Hello PDU odesílá jako DIS adresu IS `R2` na základě jeho vyšší MAC adresy.

4. Ke stejné situaci dochází i na IS `R3`. Jeho sousedství s `R2` přechází do stavu `Report`.

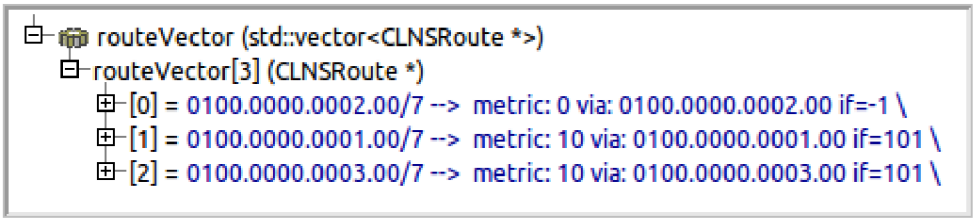
5. A následná Hello zpráva už obsahuje `R2` jako DIS na základě vyšší priority.

V kódu 5.4 je obsah LSP databáze směrovače R2 v čase 8,413 s. Obsahuje celkem čtyři LSP, všechny s *Fragment-ID* 0.

Kód 5.4: Obsah LSP databáze směrovači R2 v OMNeT++.

```
1  L1 LSP database of IS 49.0001.0100.0000.0002.00
2  No. of records in database: 4
3          0100.0000.0001.00-00    0x00000001      00199
4                  0100.0000.0002.01       metric: 10
5          0100.0000.0002.00-00    0x00000001      00200
6                  0100.0000.0002.01       metric: 10
7          0100.0000.0002.01-00    0x00000001      00200
8                  0100.0000.0001.00       metric: 00
9                  0100.0000.0003.00       metric: 00
10                 0100.0000.0002.00       metric: 00
11         0100.0000.0003.00-00    0x00000001      00199
12                 0100.0000.0002.01       metric: 10
```

V čase 13.399 s jsou LSP databáze všech IS synchronizovány. CLNS směrovací tabulka,



Obrázek 5.12: CLNS tabulka na IS R2.

viz obrázekisisLANclnsTable, obsahuje cestu do všech IS v topologii, včetně cesty do samotného IS s metrikou 0.

5.3.5 Srovnání

V obou simulacích se po velmi krátkou dobu posílají Hello zprávy se *špatným* DIS. R1 i R3 by měly již po doručení první Hello zprávy zjistit, že mají nižší prioritu (R3), nebo nižší MAC adresu (R1). Ale protože do volby DIS jsou zahrnuty pouze IS, se kterými máme navázané sousedství, dojde k výměně několika Hello PDU. Z reálné komunikace nemůžeme zjistit přesný stav sousedství ve chvíli odesílání Hello zprávy, ale z výpisů simulace ano. A protože víme, že přechod sousedství do stavu Up vyvolá generování LSP, můžeme i z reálné komunikace odhadnout tento moment.

Po úspěšném navázání sousedství už v obou simulacích dochází ke správné volbě DIS.

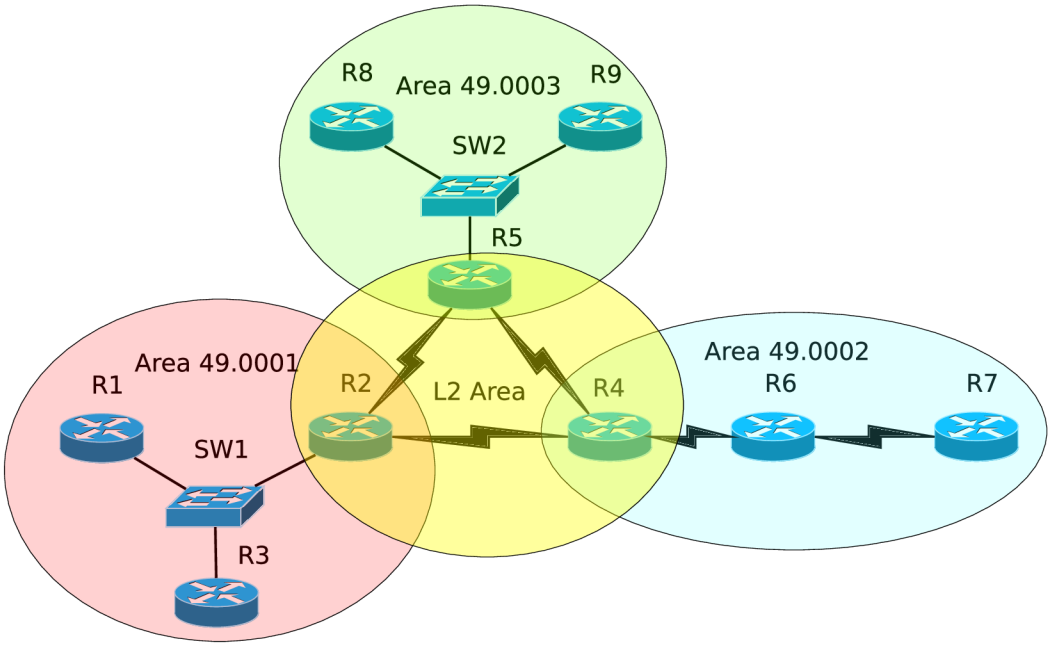
LSP databáze v obou topologiích se shodují včetně metrik z a do *pseudonode*.

5.4 IS-IS - komplexní topologie

V této ukázce nebudeme používat zachycenou komunikaci na jednotlivých linkách, ale ukážeme si na příkladech vytváření směrovací tabulky. Popisování komunikace na jednotlivých linkách by bylo zdlouhavé, a především vychází z předchozích příkladů.

5.4.1 Topologie

Struktura topologie je na obrázku 5.13. Tvoří ji tři *Level 1* oblasti `49.0001`, `49.0002` a `49.0003`. Tři IS `R2`, `R4` a `R5` tvoří páteřní *Level 2* oblast propojující jednotlivé *Level 1* oblasti.



Obrázek 5.13: Komplexní topologie IS-IS.

Přehled konfigurace jednotlivých IS je v tabulce 5.2. U směrovačů, které jsou připojeny na sdílený segment, je uvedena i priorita stát se DIS.

Model	Hostname	NET	Level	Priority
Cisco 2691	R1	49.0001.0100.0000.0001.00	1	100
Cisco 2691	R2	49.0001.0100.0000.0002.00	1-2	64
Cisco 2691	R3	49.0001.0100.0000.0003.00	1	100
Cisco 2691	R4	49.0002.0100.0000.0004.00	1-2	-
Cisco 2691	R5	49.0003.0100.0000.0005.00	1-2	64
Cisco 2691	R6	49.0002.0100.0000.0006.00	1	-
Cisco 2691	R7	49.0002.0100.0000.0007.00	1	-
Cisco 2691	R8	49.0003.0100.0000.0008.00	1	100
Cisco 2691	R9	49.0003.0100.0000.0009.00	1	80

Tabulka 5.2: Konfigurace ISs v ukázce provozovaní IS-IS na komplexní topologii.

5.4.2 Scénář

Po spuštění začnou všechny směrovače navazovat sousedství a synchronizují si LSP databázi dle předchozích ukázek. *Level 2* šíří ve svých LSP `ATT` příznak a do směrovací tabulky

vkládají cesty do sousedních oblastí. *Level 1* IS vytvářejí cesty do všech ostatních směrovačů v oblasti a nastaví si `attachedIS` na odpovídající *System-ID*.

5.4.3 Reálná síť

Na výpisu v kódu 5.5 si povšimněme nastaveného `ATT` bitu v LSP `R4.00-00`. IS tím signalizuje, že je připojen k *Level 2* oblasti a slouží jako výchozí brána.

Kód 5.5: Výpis link-state databáze na `R4`.

```
R4#show isis data

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R4.00-00            * 0x0000001C   0xD762        1141              1/0/0
R6.00-00              0x00000020   0x2677        946               0/0/0
R7.00-00              0x0000001E   0x42F6        697               0/0/0
```

V kódu 5.6 je výpis topologické databáze na směrovači `R4`. Pro *Level 1* obsahuje cestu do `R6` a `R4` přes stejné rozhraní, ale jinou cenou. V *Level 2* jsou `R2` a `R5`.

Kód 5.6: Výpis topologické databáze na směrovači `r4`.

```
R4#show isis topology

IS-IS paths to level-1 routers
System Id            Metric     Next-Hop             Interface   SNPA
R4                   --
R6                   10         R6                   Se0/2       *HDLC*
R7                   20         R6                   Se0/2       *HDLC*

IS-IS paths to level-2 routers
System Id            Metric     Next-Hop             Interface   SNPA
R2                   10         R2                   Se0/0       *HDLC*
R4                   --
R5                   10         R5                   Se0/1       *HDLC*
```

Kód 5.7 zobrazuje výpis CLNS směrovací tabulky na směrovači `R4`. Obsahuje adresu *Level 1* oblasti IS `R4` označené jako `connected`, a dvě adresy *Level 2* oblastí `49.0001` a `49.0003` vložené protokolem `IS-IS`.

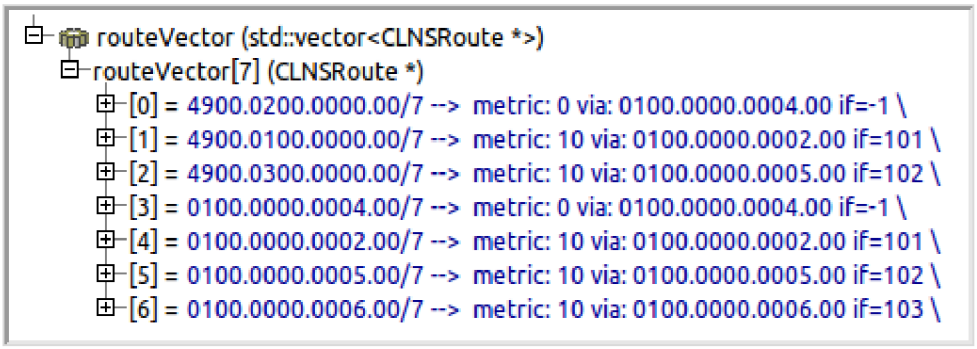
Kód 5.7: CLNS cesty do ostatních oblastí na `R4`.

```
R4#show clns route
Codes: C - connected, S~- static, d - DecnetIV
       I~- ISO-IGRP,  i~- IS-IS,  e - ES-IS
       B - BGP,       b - eBGP-neighbor

C  49.0002.0100.0000.0004.00 [1/0], Local IS-IS NET
C  49.0002 [2/0], Local IS-IS Area

i~49.0001 [110/10]
      via R2, Serial0/0
i~49.0003 [110/10]
      via R5, Serial0/1
```

5.4.4 Simulace

Pro porovnání jsme vybrali směrovací tabulku na IS `R4`. Záznamy pro dostupné oblasti jsou uvedeny jako první. Za nimi následují cesty do dostupných *L2* a *L1* IS.

Všechny *Level 1* IS mají nastavený *attached IS* na příslušný *L1L2* IS v dané oblasti. *Attached IS* má stejný význam jako výchozí brána v IP.



Obrázek 5.14: CLNS tabulka na IS `R4`.

5.4.5 Srovnání

Zatímco na Cisco směrovačích nejsou *Level 1* sousedé obsaženi v CLNS tabulce v naší simulaci ano. Pro kompletní srovnání jsme proto použili i výpis topologické databáze a sousedů.

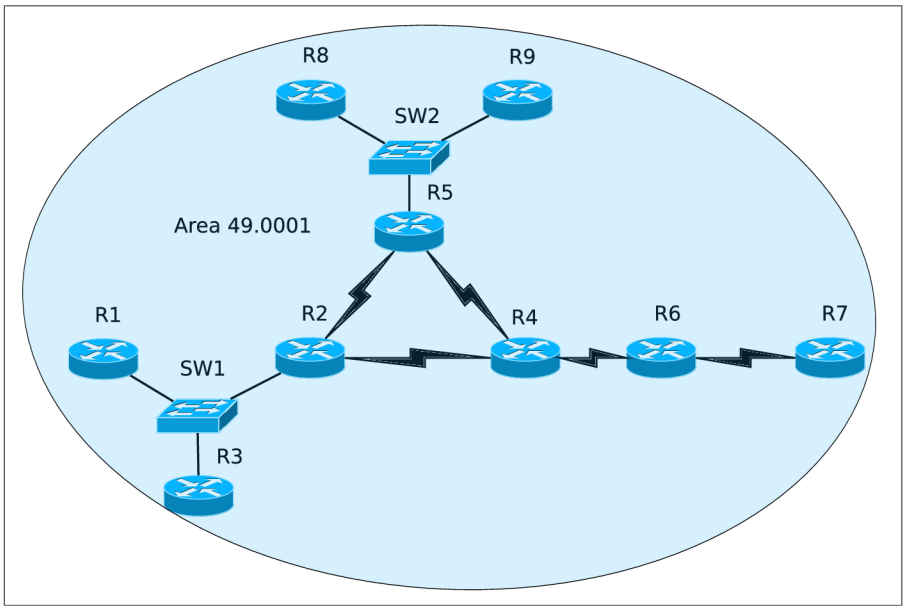
V obou případech dostáváme stejný výsledek, ale reálná síť rychleji konverguje. Cisco směrovače počítají CLNS tabulku na základě změn v LSP databázi, kdežto v simulaci se používá pevných intervalů.

5.5 IS-IS rozsáhlá oblast

V poslední ukázce pro IS-IS použijeme pouze simulační prostředí.

5.5.1 Topologie

Topologie je založena na předchozím příkladu s tím, že všechny IS jsou součástí jedné rozsáhlé *Level 1* oblasti.



Obrázek 5.15: Komplexní topologie IS-IS.

Model	Hostname	NET	Level	Priority
Cisco 2691	R1	49.0001.0100.0000.0001.00	1	100
Cisco 2691	R2	49.0001.0100.0000.0002.00	1	64
Cisco 2691	R3	49.0001.0100.0000.0003.00	1	100
Cisco 2691	R4	49.0001.0100.0000.0004.00	1	-
Cisco 2691	R5	49.0001.0100.0000.0005.00	1	64
Cisco 2691	R6	49.0001.0100.0000.0006.00	1	-
Cisco 2691	R7	49.0001.0100.0000.0007.00	1	-
Cisco 2691	R8	49.0001.0100.0000.0008.00	1	100
Cisco 2691	R9	49.0001.0100.0000.0009.00	1	80

Tabulka 5.3: Konfigurace ISs v ukázce provozovaní IS-IS na rozsáhlé oblasti.

5.5.2 Scénář

Nebudeme popisovat činnost všech IS, ale pouze těch nejvzdálenějších (`R1`, `R7` a `R8`). Budeme pozorovat jaký vliv na navazování sousedství a distribuci LSP má typ linky z hlediska

uplynutého času.

V další ukázce si předvedeme fragmentování LSP a nakonec ukázku vytvoření směrovací tabulky pro prezentaci *multipathing*.

5.5.3 Simulace

Následuje popis nejdůležitějších časových okamžiků.

- **t=2.437 - 2.672** `R1` a `R8` odesílají své LAN Hello. Protože oba předpokládají, že jsou DIS je tento interval třetinový.
- **t=5.121** `R1` má první sousedství ve stavu `Up`.
- **t=5.362** i `R8` má první úspěšně navázané sousedství.
- **t=5.407** druhé sousedství pro `R8`.
- **t=5.471** také `R1` má obě sousedství.
- **t=8.131** `R8` dostává první LSP.
- **t=8.334** `R7` odesílá první PTP Hello PDU.
- **t=8.601** `R1` dostává první LSP.
- **t=9.764** `R1` odesílá svoje LSP.
- **t=13.257** `R8` rozesílá první LSP.
- **t=17.344** `R7` přechází do stavu `Init`.
- **t=17.885** `R7` úspěšně navazuje sousedství.
- **t=22.819** `R7` dostává první LSP a protože na PTP lince se posílají všechny LSP ve frontě, dostává rovnou tři. V tuto chvíli má `R1` pět záznamu v LSP databázi a `R8` čtyři.
- **t=32.357** aktuální stav je: pět pro `R7`, sedm pro `R1` a šest pro `R8`.
- **t=133.945** `R1` má celkem 11 záznamů, ale pro dvě LSP má zatím pouze hlavičku.
- **t=137.625** `R7` má jako první kompletní databázi.
- **t=141.812** `R1` kompletuje svou LSP databázi.
- **t=143.625** `R8` jako poslední kompletuje svoji LSP databázi.

IS, které jsou na LAN segmentu rychleji navazují sousedství, protože při spuštění jednají všichni jako DIS, a posílají Hello zprávy 3x rychleji.

IS `R7` po počátečním *zaváhání* dokázal synchronizovat svoji LSP databázi nejrychleji ze sledovaných směrovačů. Rychlejší konvergenci způsobuje to, že na PTP rozhraních se posílají naráz všechny LSP, které čekají ve frontě. Na LAN se náhodně vybírá jedno z čekajících.

5.5.4 Testování fragmentace

V rámci této topologie jsme navíc provedli ještě testování funkčnosti fragmentování LSP. V rozporu se specifikací, jsme pro tuto simulaci dočasně nastavili maximální využitelnou velikost zpráv na 21 B. Do této velikosti se počítají pouze TLV položky. To způsobí, že se do každé LSP zprávy vejde TLV #2 pouze s jedním záznamem.

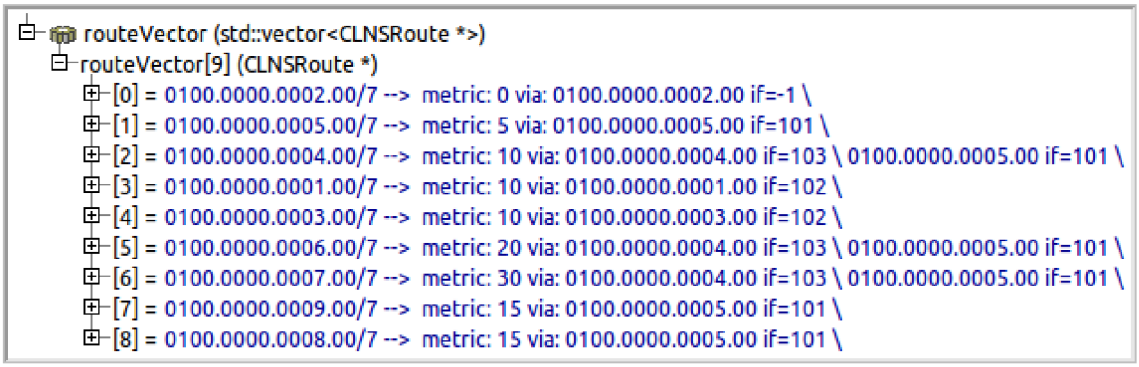
Každý IS, tak generuje několik *LSP-ID* lišících se pouze ve *Fragment-ID*.

Kompletní výpis LSP databáze za použití fragmentování je vidět v kódu 5.8.

Použití fragmentování nemělo žádný vliv na vytváření nejkratších cest a jejich vkládání do směrovací tabulky. Dvojnásobné množství LSP se na době konvergence sítě projevilo pouze minimálně. Nejrychleji měl kompletní databázi opět `R7` v čase 146.147 s, což je zpoždění menší než 9 s. Následoval `R8` v čase 146.623 s a `R1` v 149.467 s.

5.5.5 Multipathing

Abychom dokázali prezentovat podporu *multipathing*, upravili jsme výchozí metriku několika linek. Jedná se o spojení z `R2` do `R5` a z `R5` do `R4`. Metrika u těchto linek byla změněna na 5. Díky tomu se ze směrovače `R2` generují dvě možné cesty do `R4`, `R6` a `R7`.



Obrázek 5.16: CLNS tabulka na IS `R2`.

5.6 TRILL

Testování protokolu probíhalo pouze v simulačním prostředí OMNeT++, protože aktuálně není možné provozovat čistě TRILL na reálném zařízení.

Funkčnost protokolu TRILL si ukážeme na jednoduché topologii se šesti RBridge a dvěma koncovými stanicemi. Popis celé simulace začíná až po zkonvergování protokolu IS-IS, kdy v čase 100 s jedna ze stanic iniciuje zaslání ICMP Echo request. Ukázkou nejdůležitějších událostí v daných časech představíme činnost protokolu TRILL.

Všechny RBridge jsou ve výchozím nastavení. NET adresy si generují na základě rozhraní s nejvyšší MAC adresou. MAC adresy jsou generovány simulačním prostředím ve vzestupném pořadí. Designated VLAN ID je 1 a metrika všech linek je 10. Na všech rozhraních je povolen jak nativní, tak TRILL provoz.

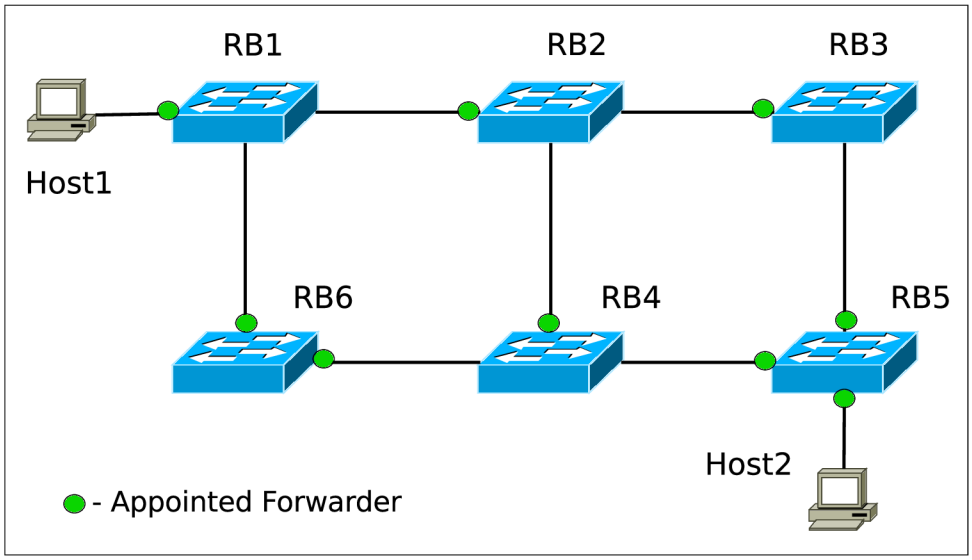
V této simulaci si popíšeme ICMP Echo Request mezi dvěma stanicemi. Stanice `Host1` má IP adresu `172.16.30.100` a stanice `Host2` `172.16.30.101` s maskou `255.255.255.0`. Obě stanice patří do VLAN ID 1.

Kód 5.8: LSP databáze na R8 při použití fragmentace.

```
L1 LSP database of IS 49.0001.0100.0000.0008.00
No. of records in database: 22
         0100.0000.0001.00-00    0x00000002      00188
                  0100.0000.0002.02       metric: 10
         0100.0000.0002.00-00    0x00000002      00191
                  0100.0000.0005.00       metric: 05
         0100.0000.0002.00-01    0x00000002      00191
                  0100.0000.0002.02       metric: 10
         0100.0000.0002.00-02    0x00000001      00191
                  0100.0000.0004.00       metric: 10
         0100.0000.0002.02-00    0x00000002      00196
                  0100.0000.0001.00       metric: 00
         0100.0000.0002.02-01    0x00000002      00187
                  0100.0000.0003.00       metric: 00
         0100.0000.0002.02-02    0x00000002      00196
                  0100.0000.0002.00       metric: 00
         0100.0000.0003.00-00    0x00000002      00176
                  0100.0000.0002.02       metric: 10
         0100.0000.0004.00-00    0x00000001      00187
                  0100.0000.0002.00       metric: 10
         0100.0000.0004.00-01    0x00000002      00179
                  0100.0000.0005.00       metric: 10
         0100.0000.0004.00-02    0x00000001      00174
                  0100.0000.0006.00       metric: 10
         0100.0000.0005.00-00    0x00000004      00199
                  0100.0000.0004.00       metric: 05
         0100.0000.0005.00-01    0x00000003      00195
                  0100.0000.0002.00       metric: 10
         0100.0000.0005.00-02    0x00000001      00188
                  0100.0000.0008.01       metric: 10
         0100.0000.0006.00-00    0x00000003      00183
                  0100.0000.0004.00       metric: 10
         0100.0000.0006.00-01    0x00000002      00187
                  0100.0000.0007.00       metric: 10
         0100.0000.0007.00-00    0x00000001      00177
                  0100.0000.0006.00       metric: 10
         0100.0000.0008.00-00    0x00000002      00190
                  0100.0000.0008.01       metric: 10
         0100.0000.0008.01-00    0x00000002      00195
                  0100.0000.0005.00       metric: 00
         0100.0000.0008.01-01    0x00000002      00190
                  0100.0000.0009.00       metric: 00
         0100.0000.0008.01-02    0x00000002      00195
                  0100.0000.0008.00       metric: 00
         0100.0000.0009.00-00    0x00000002      00197
                  0100.0000.0008.01       metric: 10
```

5.6.1 Topologie

Zelené kolečka znázorňují, který RBridge je na dané lince *Appointed Forwarder* pro VLAN ID 1. Jiné VLAN ID se v tomto příkladu nepoužívají.



Obrázek 5.17: TRILL campus topologie.

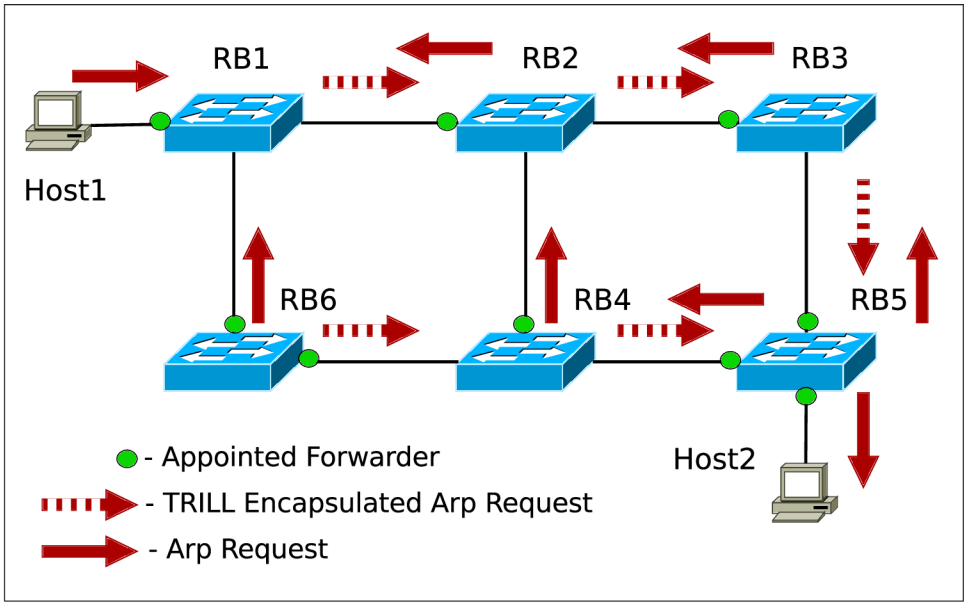
5.6.2 Scénář

Po zkonvergování protokolu IS-IS v režimu L2, pošle stanice `Host1` v čase 100 s ICMP Echo Request na IP adresu `172.16.30.101`. Protože pro ni stanice nezná MAC adresu, iniciuje zaslání ARP request na broadcast adresu `FF:FF:FF:FF:FF:FF`. Propagace ARP Request je zobrazena na obrázku 5.18. ARP Request od `Host1` jde v nativní formě a bez VLAN tagu. Na `RB1` dojde k TRILL zapouzdření a posílá se jako *multi-destination* na `RB2` a `RB6`. `RB2` jej rozbalí a zasílá v nativní podobě zpět směrem k `RB1` a TRILL zapouzdřený přeposílá na `R3`. Protože `RB1` není *Appointed Forwarder* rámec od `RB2` zahazuje. Takto postupují všechny RBridge, až TRILL rámec doputuje i na `R5`, který ho v nativní podobě zasílá stanici `Host2`. Všechny RBridge, které rámec rozbalily a zaslaly na rozhraní, kde jsou *Appointed Forwarder* se zároveň naučily výstupní RBridge pro zdrojovou MAC adresu.

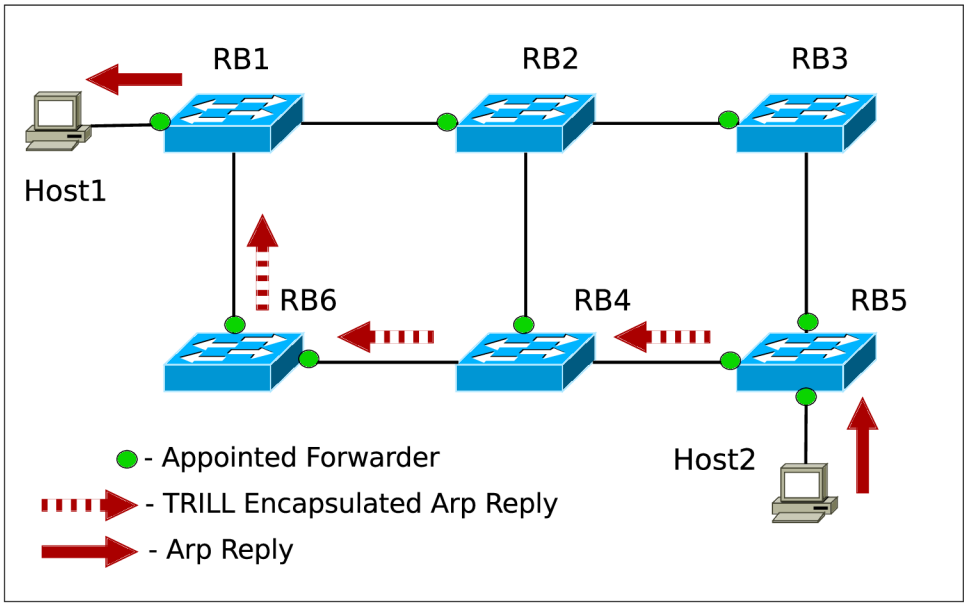
Obrázek 5.19 zobrazuje ARP reply. `Host2` vygeneruje ICMP Echo Reply a pošle jej na MAC adresu stanice `Host1` směrem k `RB5`. Ten zná výstupní RBridge pro cílovou MAC adresu. Rámec TRILL zapouzdří a posílá na `RB4` - `RB6` - `RB1`. `RB1` jako cílový RBridge rámec rozbalí, prohledá tabulku koncových stanic a zasílá směrem k `Host1`. Z rozbaleného rámce se naučí zdrojovou MAC adresu a vstupní RBridge.

`Host1` po obdržení MAC adresy pro IP adresu `172.16.30.101` odesílá ICMP Echo Request. `RB1` zjistí, že pro danou adresu zná výstupní RBridge, tak jej zapouzdří a posílá na `RB5` jak je ukazuje obrázek 5.20.

`RB5` má při zasílání ICMP Echo Reply na výběr ze dvou cest. Zvolí cestu ve směru `RB4` jak můžeme vidět na obrázku 5.21.



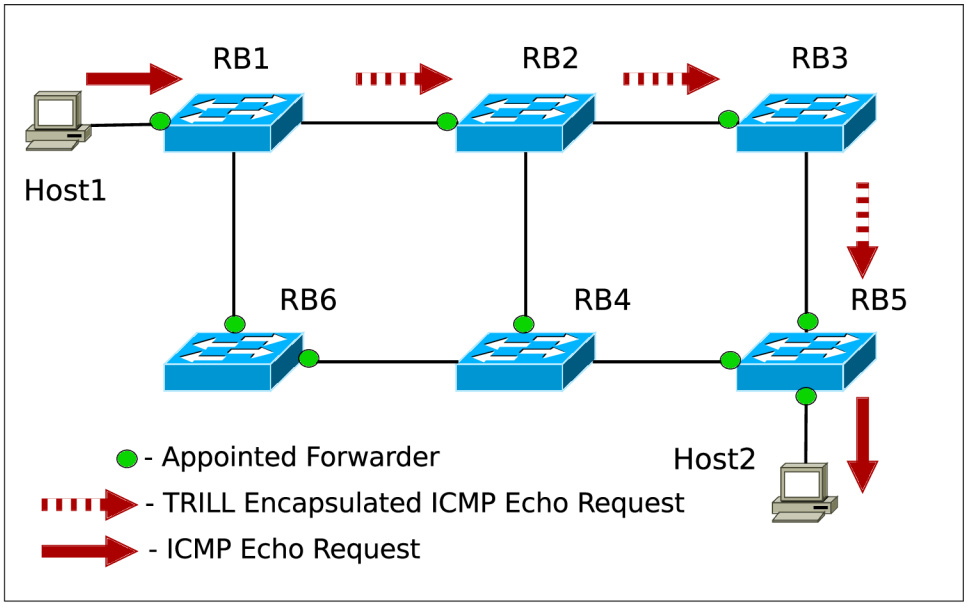
Obrázek 5.18: Propagace ARP request.



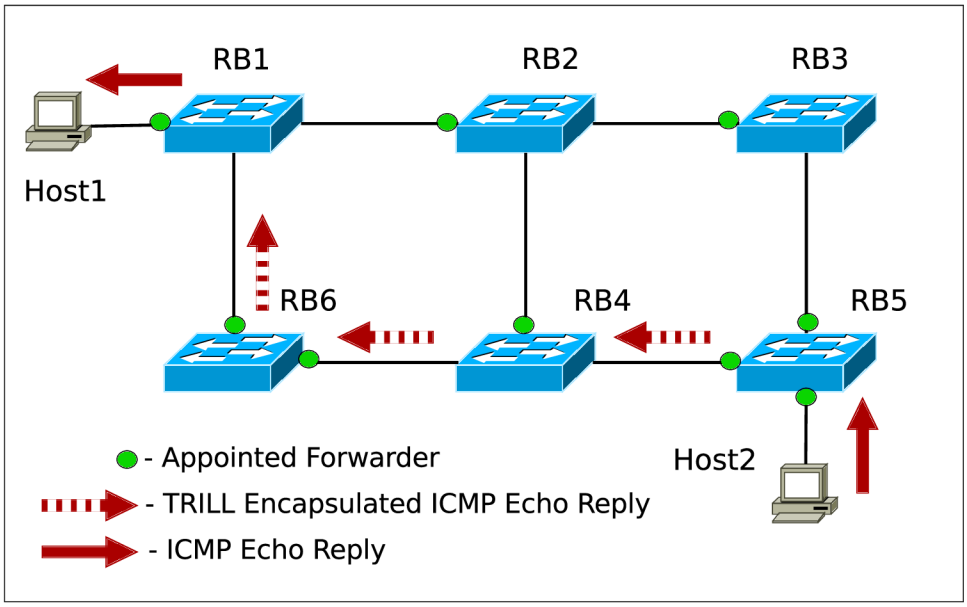
Obrázek 5.19: Propagace ARP reply.

5.6.3 Vyhodnocení

Při ARP Request je vidět, že se posílá nativní rámec zpět ke zdroji. Hned na první pohled to vypadá neefektivně a hlavně jako chyba.



Obrázek 5.20: Propagace ICMP echo request.



Obrázek 5.21: Propagace ICMP echo reply.

Toto neefektivní rozesílání je zapříčiněno dvěma fakty:

1. RBridge používá výchozí nastavení, a má tak na všech portech povolen nativní provoz.
2. Každá linka reprezentuje $0 - N$ koncových stanic, opakovačů, přepínačů apod.

Protože je na lince povolen nativní provoz, musí se na ní objevit rámce i v nativní podobě. Musíme brát v úvahu, že se nemusí jednat o přímé propojení. Pokud bychom měli reálnou

síť přesně korespondující s vyobrazenou topologií, nastavili bychom takové linky jako *point-to-point*. Na *point-to-point* linkách je totiž nativní provoz zakázaný.

Z obrázku 5.20 a 5.21 je vidět, že se v obou směrech využívá jiná cesta. Fakt, že k tomu v této ukázce dochází je sice náhodný, ale jasně demonstruje výhodu protokolu TRILL v podobě využívání více cest - multipathing. Aby to nevyznělo příliš amatérsky, směrovací tabulka podporuje multipathing, ale TRILL implementace ho aktivně nevyužívá. Pokud dostane na výběr více cest, vybere si první možnou.

V případě použití protokolu STP, by v nejlepším případě došlo k ustanovení jedné cesty pro komunikaci v obou směrech. V horším případě by mohlo dojít k vytvoření neoptimální cesty. Konvergování STP topologie do požadovaného stavu můžeme ovlivnit ruční konfigurací, ale RBridge topologie to dokáže i bez dalších zásahů.

5.7 Shrnutí

Pomocí uvedených ukázek jsme si ověřili fungování protokolu IS-IS s implementací na reálných směrovačích firmy Cisco.

Protokol TRILL jsme neměli s čím ověřovat protože aktuálně neexistuje zařízení, které by jej podporovalo. Možností by bylo provozovat protokol FabricPath, ale ten funguje pouze na přepínačích Cisco Nexus, které nejsou v rámci školních laboratoří dostupné.

Proto jsme si alespoň ukázali jakým způsobem probíhá přeposílání broadcast a známého unicast provozu za využití *multipathing*.

Kompletní konfigurační soubory všech reálných směrovačů z uvedených testovacích topologií se nachází na přiloženém CD. Společně s nimi jsou uloženy popisované záznamy komunikace ve formě `.cap` souborů.

Kapitola 6

Závěr

V této práci jsme se zabývali směrovacím protokolem IS-IS pro použití jako L3 směrovacího protokolu a také jeho rozšířením pro podporu protokolu TRILL. Seznámili jsme se se stavem podpory OSI protokolů v nástroji OMNeT++ s knihovnami INET a ANSAINET.

Cílem práce bylo implementovat protokol IS-IS a TRILL v nástroji OMNeT++, rozšířit tak pole podporovaných protokolů a provést zajímavé simulace. L3 verze protokolu IS-IS byla implementována a ověřena na sérii porovnání reálných sítí se simulacemi. Implementace protokolu TRILL spolu s L2 IS-IS byla otestována pouze v simulačním prostředí, protože nemáme přístup k zařízení, které by dané protokoly podporovalo.

6.1 Vlastní přínos

Nejprve jsme prostudovali specifikaci protokolu IS-IS v ISO 10589:2002 a v [9] a seznámili se s jeho principy. Následně jsme prozkoumali prostředí OMNeT++ a rozšiřující knihovny INET a ANSAINET, která je vyvíjená v rámci projektu ANSA na FIT.

Poté jsem se seznámili s aktuálním stavem implementace IS-IS, která je součástí ANSAINET. V rámci daného stavu fungovalo navazování sousedství na LAN pro *Level 1* i *Level 2* oblasti a generování LSP pro *Level 1*. Zbývalo už tedy pouze dodělat generování LSP pro *Level 2* a počítání nejkratších cest.

Při podrobnějším studiu jsme zjistili, že pro navázání sousedství není provedena kontrola obousměrnosti a volba DIS neprobíhala korektně. Zajištění rozesílání Hello pro DIS ve třetinových intervalech bylo řešeno globální proměnnou. Generování a rozesílání LSP neodpovídalo používání SSN a SRM příznaků.

Opravili jsme tedy navazování sousedství na LAN společně s volbou DIS a přidali vytváření sousedství na point-to-point rozhraních. Mechanismus generování LSP byl rozšířen o podporu *Level 2* a generování fragmentů. Do rozesílání na broadcast jsme přidali náhodný výběr čekajících LSP. Podle specifikace jsme implementovali algoritmus SPF pro vytváření nejkratších cest. Pro jejich reprezentaci pak vytvořili také směrovací tabulku pro CLNS.

Další postup spočíval v nastudování protokolu TRILL. Poté jsme nastudovali a do aktuální verze knihovny INET 2.0 naportovali existující model přepínače. Rozšířením o mudul IS-IS, CLNS tabulky a tabulky rozhraní jsem zkonstruovali model RBridge.

Výsledkem jsou dva modely zařízení:

- Směrovač, který provozuje protokol IS-IS jako L3 směrovací protokol.
- RBridge v rámci kterého běží protokol IS-IS ve verzi L2 a protokol TRILL.

Výsledky této práce byly prezentovány také jako příspěvek [18] na studentské konferenci STUDENT EEICT 2013.

6.2 Další vývoj

Vytvořené třídy jsou součástí knihovny ANSAINET. Oba protokoly mají mnoho volitelných funkcí a IS-IS i spoustu dalších rozšíření. Proto je předpoklad, že stávající implementace může sloužit jako základ pro vytváření dalších projektů.

Možnosti dalšího vývoje jsme zmínili v sekci 4.15. Především pak rozšíření o podporu IPv4 by otvíralo široké možnosti využití.

Pro protokol TRILL pak zavedení *wide-metric* TLV pro lepší rozlišení rychlosti linky při automatické konfiguraci metrik. Protože se v rámci projektu ANSA významně pracuje na podpoře multicastu bylo by vhodné rozšířit protokol TRILL i o zpracování zpráv protokolů zajišťujících multicast.

Literatura

[1] Cisco IOS IP Routing: ISIS Command Reference. [online], November 2010 [cit. 2012-03-02].
URL http://www.cisco.com/en/US/docs/ios/iproute_isis/command/reference/irs_cr_book.pdf

[2] Cisco FabricPath. [online], 2010 [cit. 2012-10-23].
URL www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9402/at_a_glance_c45-605626.pdf

[3] Banerjee, A.; Ward, D.: Extension to IS-IS for Layer-2 Systems. [online], April 2011 [cit. 2013-01-18].
URL http://tools.ietf.org/html/rfc6165

[4] Callon, R.: Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. [online], December 1990 [cit. 2012-03-02].
URL http://tools.ietf.org/html/rfc1195

[5] Doyle, J.: *OSPF and IS-IS: Choosing an IGP for Large-Scale Networks*. Addison Wesley Professional, 2005, ISBN 0-321-16879-8.
URL http://fengnet.com/book/OSPFandISIS/main.html

[6] Eastlake, D.; Banerjee, A.; Dutt, D.; aj.: Transparent Interconnection of Lots of Links (TRILL) Use of IS-IS. [online], July 2011 [cit. 2012-10-07].
URL http://tools.ietf.org/html/rfc6326

[7] Eastlake, D.; Perlman, R.; Ghanwani, A.; aj.: Routing Bridges (RBridges): Adjacency. [online], July 2011 [cit. 2012-10-07].
URL http://tools.ietf.org/html/rfc6327

[8] Eastlake, D.; Schiller, J.; Crocker, S.: Randomness Requirements for Security. [online], June 2005 [cit. 2013-03-22].
URL http://tools.ietf.org/html/rfc4086

[9] Gredler, H.; Goralski, W.: *The Complete IS-IS Routing Protocol*. Springer, 2004, ISBN 9781852338220.

[10] Hrnčiřík, M.: *Modelování L2 protokolů zajišťujících bezsmyčkovost*. diplomová práce, FIT VUT v Brně, Brno, 2012.

[11] ISO/IEC: 8473-1:1998: Information technology — Protocol for providing the connectionless-mode network service: Protocol specification. For review, International Organization for Standardization, Geneva, Switzerland., 5. March 1998

[cit. 2013-04-10].
URL http://standards.iso.org/ittf/PubliclyAvailableStandards/c030931_ISO_IEC_8473-1_1998(E).zip

[12] ISO/IEC: 9542:1998: Information processing systems — Telecommunications and information exchange between systems — End system to Intermediate system routeing exchange protocol for use in conjunction with the Protocol for providing the connectionless-mode network service (ISO 8473). For review, International Organization for Standardization, Geneva, Switzerland., 15. August 1998 [cit. 2013-04-10].
URL http://standards.iso.org/ittf/PubliclyAvailableStandards/s017285_ISO_IEC_9542_1988(E).zip

[13] ISO/IEC: 10589:2002: Information technology — Telecommunications and information exchange between systems — Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473) . For review, International Organization for Standardization, Geneva, Switzerland., 15. October 2002 [cit. 2012-03-02].
URL http://standards.iso.org/ittf/PubliclyAvailableStandards/c030932_ISO_IEC_10589_2002(E).zip

[14] ISO/IEC: 8348:2002: Information technology — Open Systems Interconnection — Network service definition. For review, International Organization for Standardization, Geneva, Switzerland., 1. November 2002 [cit. 2012-11-02].
URL http://standards.iso.org/ittf/PubliclyAvailableStandards/c035872_ISO_IEC_8348_2002(E).zip

[15] ITU-T: Information technology — Protocol for providing the connectionless-mode network service: Protocol specification. Recommendation X.233, International Telecommunication Union, Geneva, 1997.

[16] Katz, D.; Saluja, R.; Eastlake, D.: Three-Way Handshake for IS-IS Point-to-Point Adjacencies. [online], October 2008 [cit. 2012-10-07].
URL http://tools.ietf.org/html/rfc5303

[17] Li, T.; Ginsberg, L.: IS-IS Registry Extension for Purges. [online], May 2011 [cit. 2012-10-07].
URL http://tools.ietf.org/html/rfc6233

[18] Marek, M.: Modelling of IS-IS and TRILL. In *Proceedings of the 19th conference STUDENT EEICT 2013*, FIT VUT v Brně, 2013, ISBN 978-80-214-4694-6, s. 222–224.

[19] Martey, A.; Sturgess, S.: *IS-IS Network Design Solutions*. Networking Technology Series, Cisco Press, 2002, ISBN 9781578702206.

[20] Oran, D.: OSI IS-IS Intra-domain Routing Protocol. [online], February 1990 [cit. 2012-03-02].
URL http://tools.ietf.org/html/rfc1142

[21] Parker, J.: Recommendations for Interoperable Networks using Intermediate System to Intermediate System (IS-IS). [online], February 2004 [cit. 2012-03-02].
URL http://tools.ietf.org/html/rfc3719

[22] Perlman, R.; Eastlake, D.; Gai, D. S.; aj.: Routing Bridges (RBridges): Base Protocol Specification. [online], July 2011 [cit. 2012-10-07].
URL http://tools.ietf.org/html/rfc6325

[23] Rekhter, Y.; Moskowitz, B.; Karrenberg, D.; aj.: Address Allocation for Private Internets. [online], February 1996 [cit. 2012-03-02].
URL http://tools.ietf.org/html/rfc1918

[24] Shen, N.; Zinin, A.: Point-to-Point Operation over LAN in Link State Routing Protocols. [online], October 2008 [cit. 2012-10-07].
URL http://tools.ietf.org/html/rfc5309

[25] Vasseur, J.; Shen, N.; Aggarwal, R.: Intermediate System to Intermediate System (IS-IS) Extensions for Advertising Router Information. [online], July 2007 [cit. 2013-04-07].
URL http://tools.ietf.org/html/rfc4971

Příloha A

Obsah CD

Přiložené CD obsahuje následující soubory a složky:

- Elektronická verze textu diplomové práce ve formátu PDF.
- Adresář `tex` se zdrojový textem pro v LaTeXu pro vytvoření této práce.
- Adresář `src` se zdrojovými kódy knihovny ANSAINET i s vytvořenými soubory.
- Adresář `doc` obsahující programovou dokumentaci vytvořenou pomocí nástroje doxygen.
- Adresář `examples` obsahuje příklady simulací na kterých byla implementace testována.
- Adresář `configs` s konfiguračními soubory směrovačů z ukázek pro reálné sítě.
- Adresář `capture` se soubory zachycené komunikace popisované v kapitole testování.

Příloha B

Seznam použitých zkratek

- **ISO** – **I**nternational **O**rganization for **S**tandardization
- **IETF** – **I**nternet **E**ngineering **T**ask **F**orce
- **ITU-T** – **I**nternational **T**elecommunication **U**nion - **T**elecommunication Standardization Sector
- **OSI** – **O**pen **S**ystem **I**nterconnection označuje snažení ISO pro standardizaci síťových protokolů
- **NET** – **N**etwork **E**ntity **T**itle.
- **OSPF** – **O**pen **S**hortest **P**ath **F**irst je směrovací protokol.
- **BGP** – **B**order **G**ateway **P**rotocol
- **TRILL** – **T**ransparent **I**nterconnection of **L**ots of **L**inks
- **IS-IS** – **I**ntermediate **S**ystem to **I**ntermediate **S**ystem
- **ES-IS** – **E**nd **S**ystem to **I**ntermediate **S**ystem
- **IPv4** – **I**nternet **P**rotocol **v**ersion **4**
- **IPv6** – **I**nternet **P**rotocol **v**ersion **6**
- **AFI** – **A**ddress **F**amily **I**dentifier
- **IIH** – **I**S-**I**S **H**ello
- **LAN** – **L**ocal **A**rea **N**etwork
- **PTP** – **P**oint-**t**o-**p**oint
- **DIS** – **D**esignated **I**ntermediate **S**ystem
- **DRB** – **D**esignated **R**outing **B**ridge
- **AF** – **A**ppointed **F**orwarder
- **PDU** – **P**rotocol **D**ata **U**nit

- **CSNP** – **C**omplete **S**equence **N**umber **P**DU
- **PSNP** – **P**artial **S**equence **N**umber **P**DU
- **MTU** – **M**aximum **T**ransmission **U**nit
- **VLAN** – **V**irtual **LAN**
- **MAC** – **M**edia **A**ccess **C**ontrol
- **TLV** – **T**ype **L**ength **V**alue
- **TTL** – **T**ime **T**o **L**ive
- **OMNeT++** – **O**bjective **M**odular **N**etwork **T**estbed in C++
- **NED** – **N**etwork **De**scription je formát souboru popisující síťovou topologii v OMNeT++
- **ESADI** – **E**nd **S**ystem **A**ddress **D**istribution **I**nformation
- **STP** – **S**panning **T**ree **P**rotocol
- **NLPID** – **N**etwork **L**ayer **P**rocotol **I**dentifier
- **XML** – E**x**tensible **M**arkup **L**anguage
- **JUNOS** – je operační systém používaný na sítových zařízeních firmy Juniper
- **SRM** – **S**end **R**outeing **M**essage
- **SSN** – **S**end **S**equence **N**umbers
- **CLNS** – **C**onnectionless-mode **N**etwork **S**ervice
- ***L2*** – *Level 2* označuje úroveň oblasti protokolu IS-IS
- **L2** – **Layer 2** označuje vrstvu v rámci síťového modelu ISO/OSI
- **SPF** – **S**hortest **P**ath **First** je algoritmus pro počítání nejkratších cest v grafu
- **PDU** – **P**rotocol **D**ata **Unit** je obecné označení jednotky používané na dané vrstvě
- **IOS** – **I**nternetwork **O**perating **S**ystem
- **PDF** – **P**ortable **D**ata **F**ormat je přenosný formát dokumentů vytvořený firmou Adobe
- **RFC** – **R**equest **F**or **Comments** se používá pro označení řady standardů popisujících internetové protokoly, systémy, apod.

Příloha C

TLV a sub-TLV

IS-IS je prezentovaný jako výborný směrovací protokol díky svojí jednoduché rozšiřitelnosti. V této kapitole si popíšeme mechanismus, pomocí kterého je ona jednoduchá rozšiřitelnost dosažena. Jedná se o položky TLV *Type*, *Length*, *Value*. Mohou přenášet libovolnou informaci a jediným pevným omezením je minimální délka 2 B a maximální 257 B včetně hlavičky. Vždy musíme přenášet alespoň typ a délku, i kdyby byla 0. A protože velikost položky *Length* je 1 B, dokáže vyjádřit maximálně hodnotu 255 jako délku pole *Value*. V položce *Length* je pouze velikost *Value*. Syntaxe a sémantika pole *Value* záleží čistě na typu TLV.

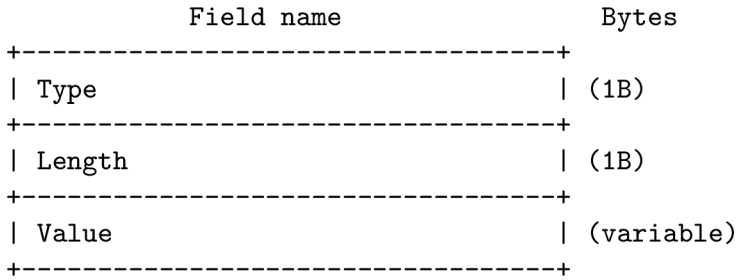
Flexibilita TLV přináší zvýšenou komplexitu do jejich zpracování.

C.1 sub-TLV

Protože rozsah typů je omezený velikostí pole *Type* 1 B byly zavedeny *sub-TLV* položky. Místo původních 256 typů, máme nyní teoreticky k dispozici až 65536, pokud neuvažujeme již definované a rezervované typy. Představený mechanismus neomezuje definování sub-TLV v jiném sub-TLV, jejich množství tak není nijak limitované. Při návrhu nových sub-TLV musí být brána v úvahu přidaná režije hlavičky (*Type* a *Length*). Nevyplatí se vytvářet příliš krátké sub-TLV. Sub-TLV mají stejný formát jako TLV, ale jejich platnost není omezená pouze na výskyt v určitých typech zpráv, ale i konkrétních TLV.

C.2 Formátování

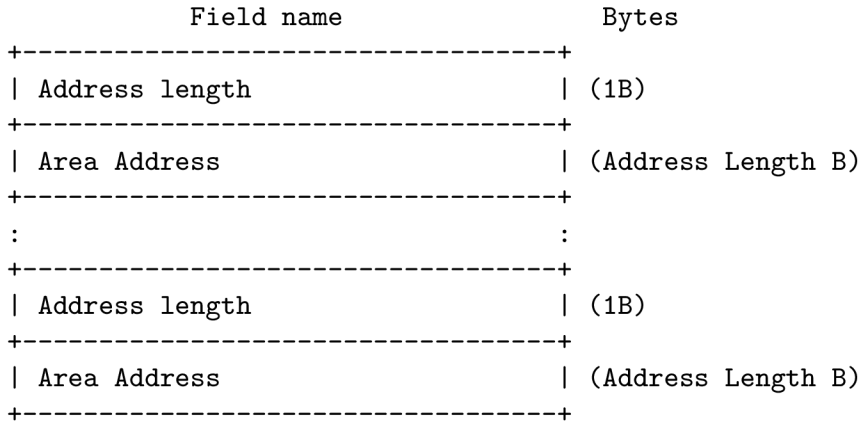
Na obrázku C.1 je formát hlavičky TLV.



Obrázek C.1: Formát TLV a sub-TLV

C.3 Area Addresses

Area Addresses TLV #1, nebo také *manualAreaAddresses* obsahuje adresy oblastí, do kterých daný IS patří. Jednotlivé adresy mohou mít různou délku. Používá se v IS-IS Hello, TRILL Hello i v LSP zprávách. Předchází ostatním TLV položkám a může se objevit opakovaně. Nemělo by se objevit v žádných nenulových (*Fragment-ID* jiné než 0) LSP a v žádných LSP, které jsou generovány ve jménu pseudonodu. Formát TLV #1 je na obrázku C.2.



Obrázek C.2: Formát TLV #1 Area Addresses

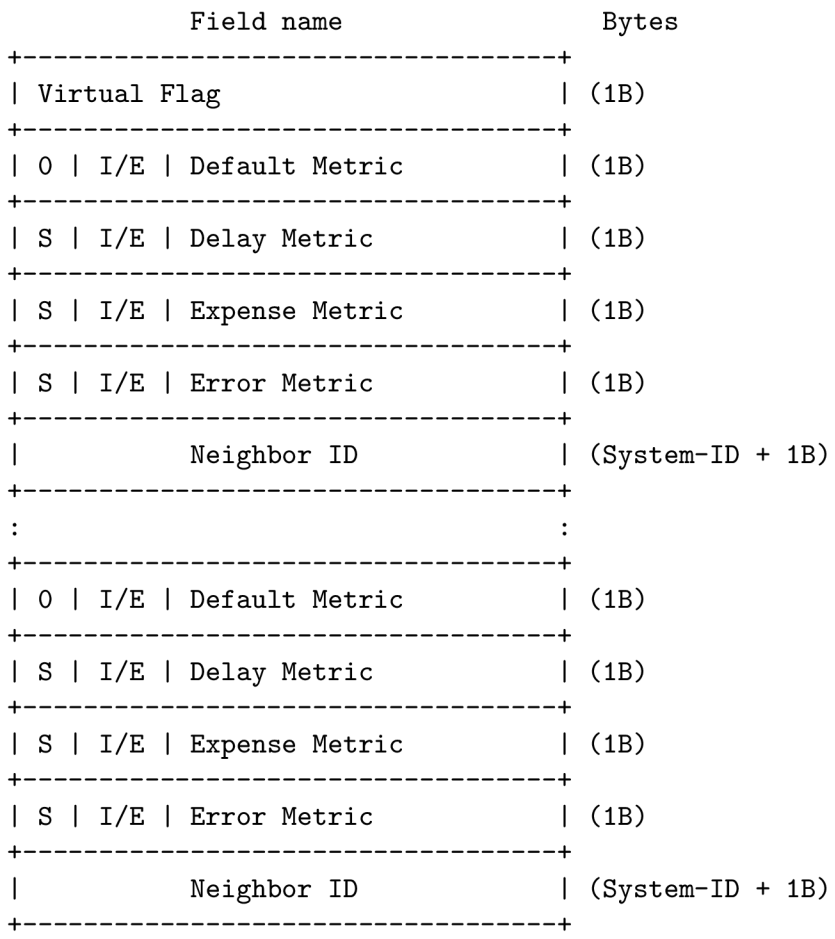
V TRILL Hello zprávách má pevně daný formát. Obsahuje pouze jednu adresu délky 1 B s hodnotou 0.

C.4 Intermediate System Neighbors

Intermediate System Neighbours TLV#2 se může objevit více než jednou v LSP s jakýmkoliv *Fragment-ID*. Délka je v násobcích $1 + N * (\textit{System-ID} + 5)$, kde N je počet uvedených sousedů a *System-ID* délka této položky. Formát TLV #2 je na obrázku C.3.

- **Virtual Flag** je boolovská proměnná. Pokud je nastavena, znamená, že se ve skutečnosti jedná o linku druhé úrovně (*Level 2*) pro opravu rozdělené oblasti. Při normální činnosti ji *Level 1* IS vždy nastavují na nulu.
- **I/E** je jednobitový příznak, určující, zda daná metrika je interní, nebo externí.
- **S** nastaví IS v případě, že danou metriku nepodporuje.
- **Default Metric** je výchozí metrika k danému sousedovi.
- **Delay Metric** je hodnota metriky zpoždění linky k udanému sousedovi.
- **Expense Metric** je hodnota nákladů při použití linky k udanému sousedovi.
- **Error Metric** je chybová metrika k danému sousedovi.
- **Neighbor ID** je *LAN-ID* reportovanhé souseda.

Všechny metriky jsou 6 b neznaménková čísla. V praxi se využívá pouze výchozí metrika. Protože ostatní metriky se nevyužívají a 6 b pro udání metriky je příliš malý rozsah, bylo TLV #2 nahrazeno novým Extended IS Reachability TLV #22.



Obrázek C.3: Formát TLV #2 Intermediate System Neighbors

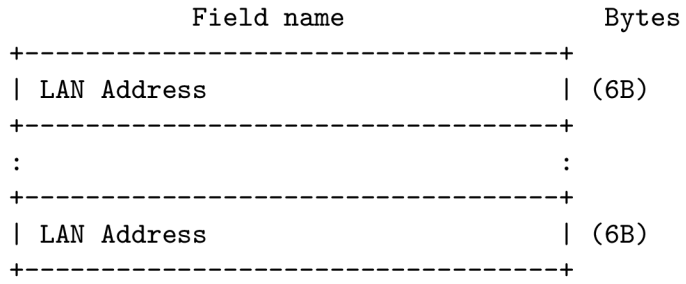
C.5 Intermediate System Neighbors

Intermediate System Neighbours TLV#6 má stejný název, ale jiný kód, a nepoužívá se v LSP, ale v LAN Hello. Má zcela odlišný formát viz obrázek C.4. Může se objevit několikrát. Do jednoho TLV se vejde až 42 záznamů $(255/6 = 42,5)$.

- **LAN Address** je MAC adresa souseda, jehož LAN Hello daný IS přijal a má pro něj sousedství ve stavu *Up*, nebo *Initialising*.

C.6 Padding

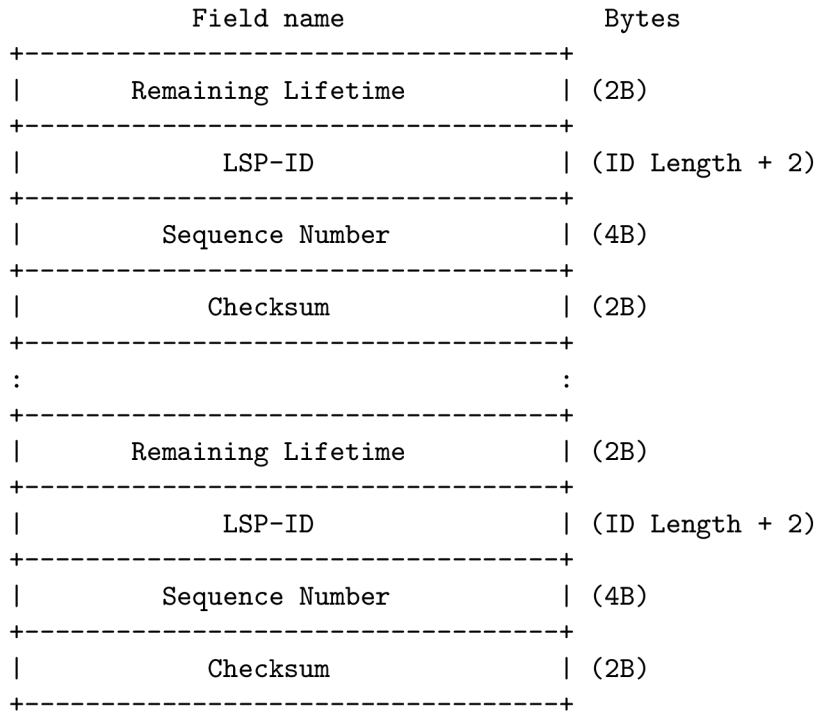
Pomocí *Padding* TLV#8 se uměle zvětšují Hello zprávy při testování MTU na dané lince. Délka může být nulová, ale včetně hlavičky, není možné vytvořit *Padding* TLV menší než 2 B. Obsah se nijak dále nezpracovává a měl by být nastaven na nulu.



Obrázek C.4: Formát TLV #6 Intermediate System Neighbors

C.7 LSP Entries

LSP Entries TLV#9 se používají v CSNP a PSNP pro synchronizaci LSP databáze. Může se použít vicekrát v jedné zprávě za předpokladu, že jednotlivé záznamy jsou vzestupně seřazeny podle *LSP-ID*. Posloupnost musí být dodržena v rámci jednoho TLV i mezi jeho dalšími výskyty.



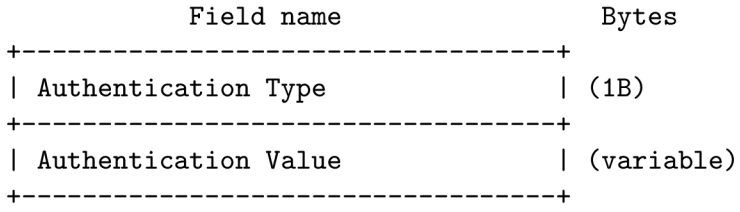
Obrázek C.5: Formát TLV #9 LSP Entry

- **Remaining Lifetime** určuje dobu, za kterou tento záznam vyexpiruje a bude zahájeno vymazání z LSP databáze.
- **LSP-ID** identifikuje IS, pro který je tato položka určena.
- **Sequence Number** specifikuje verzi pro dané *LSP-ID*. Vyšší hodnota znamená novější verzi.

- **Checksum** kontrolní součet *Remaining Lifetime*, *LSP-ID* a *Sequence Number*.

C.8 Authentication Information

Authentication Information TLV#10 volitelně zajišťuje autentizaci původce daného PDU. Může se použít ve všech typech PDU.



Obrázek C.6: Formát TLV #10 Authentication Information

- **Authentication Type** specifikuje typ použité autentizace.
- **Authentication Value** hodnota i délka tohoto pole závisí na použitém typu autentizace.

Přehled vybraných TLV je zahrnut do tabulky C.1. Pro každé TLV je uveden jeho typ, název, v jakých zprávách se používá, zda je TLV implementováno a reference, kde je TLV definováno. Sloupec **T** značí TRILL Hello a sloupec **M** určuje, zda se dané TLV, nebo sub-TLV může vyskytovat opakovaně v jedné zprávě.

Kód	Název	IIH	LSP	SNP	T	M	Imp.	Ref
1	Area Addresses	A	A	N	A	A	A	[13]
2	IIS Neighbors	N	A	N	N	A	A	[13]
6	IIS Neighbors	A	N	N	N	A	A	[13]
8	Padding	A	N	N	N	A	A	[13]
9	LSP Entries	N	N	A	N	A	A	[13]
10	Authentication Information	A	A	A	A	N	N	[13], [17]
143	MTA Port Capability	N	N	N	A	A	A	[3]
143.1	Special VLANs and Flags	N	N	N	A	A	A	[6]
143.3	Appointed Forwarders	N	N	N	A	A	A	[6]
147	MAC-Reachability	N	A	N	N	A	N	[3]
145	TRILL Neighbor	N	N	N	A	A	A	[6]
240	P2P 3-Way Adjacency State	A	N	N	N	A	A	[16]
242	IS-IS Router CAPABILITY	N	A	N	N	A	N	[25]
242.6	Nickname	N	A	N	N	A	N	[6]

Tabulka C.1: Přehled TLV položek

Příloha D

Konfigurační soubor

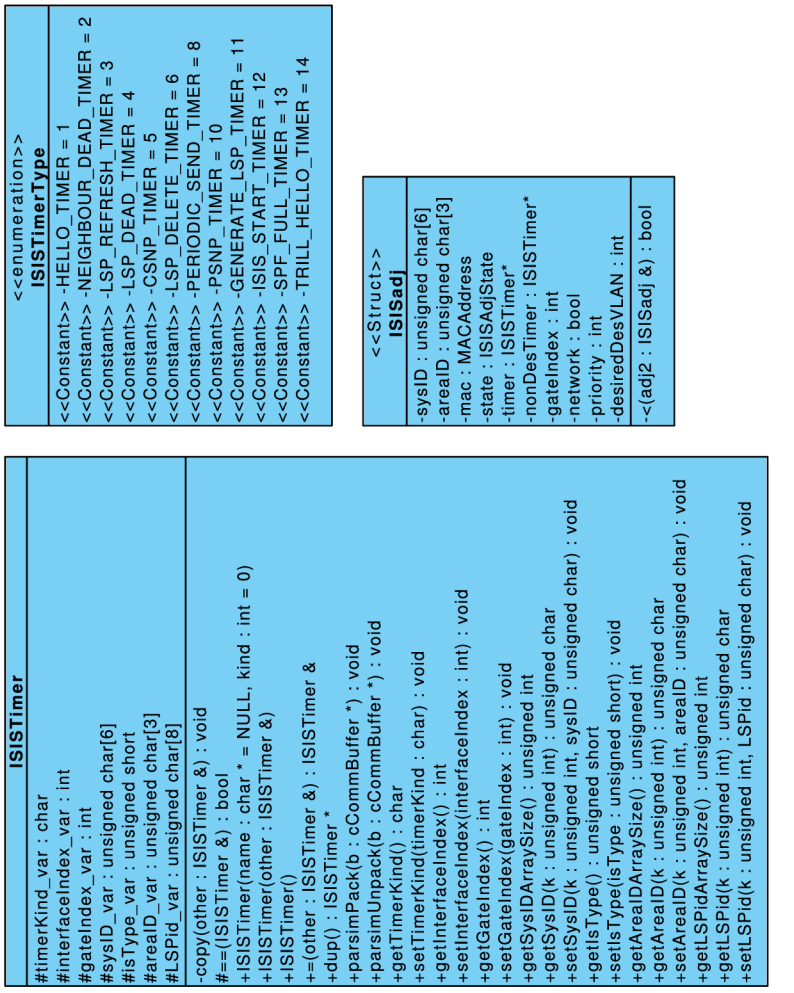
Kód D.1: Šablona konfigurovatelných parametrů protokolu IS-IS

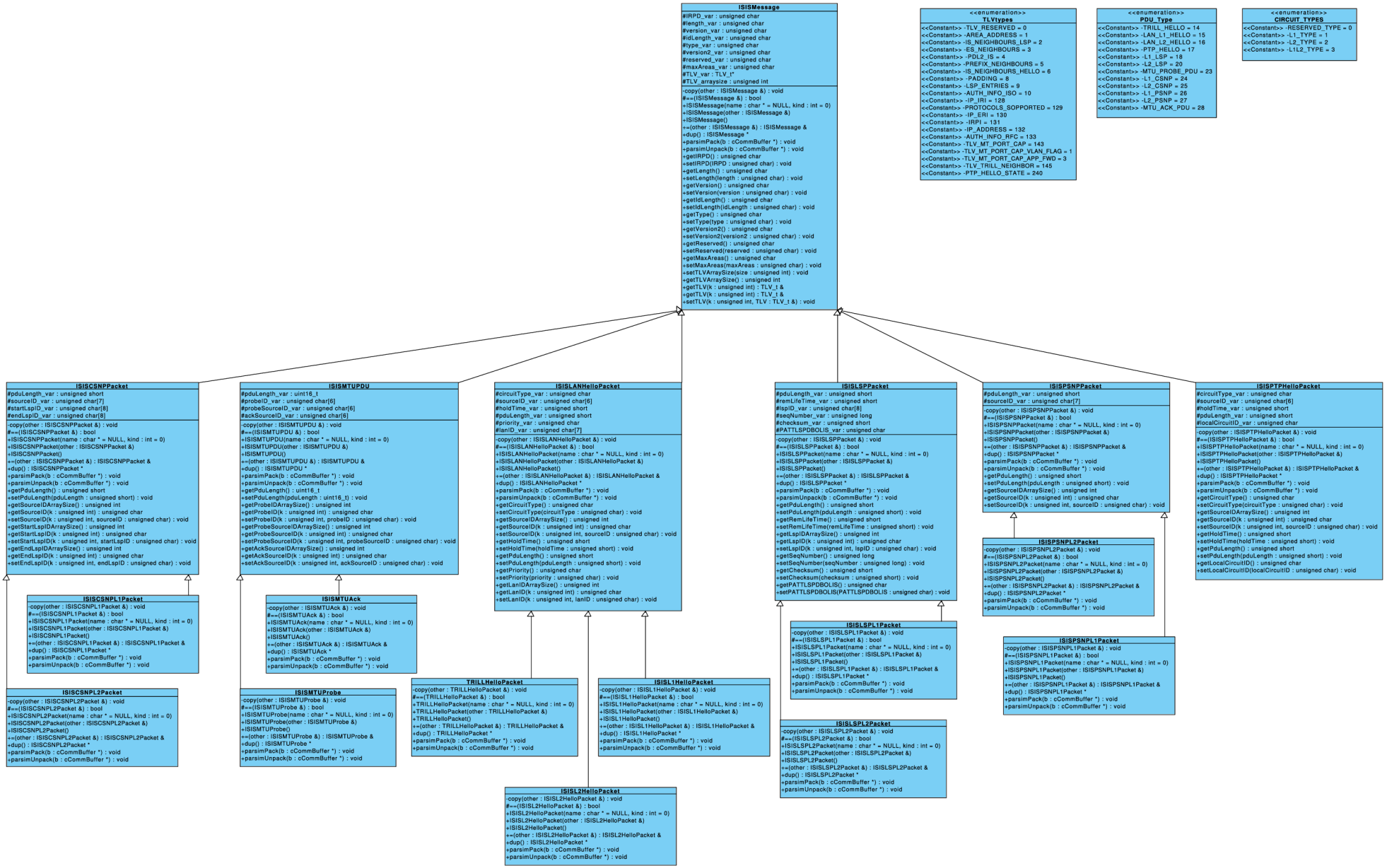
```
<Router id="IS-A">
    <Interfaces>
        <Interface name="eth0">
            <ISIS-Priority>100</ISIS-Priority>
            <ISIS-Network>broadcast</ISIS-Network> <!-- point-to-point -->
            <ISIS-Circuit-Type>L1</ISIS-Circuit-Type>  //L2, L1L2
            <ISIS-Metric>10</ISIS-Metric>
            <ISIS-Priority>64</ISIS-Priority>
            <L1-Hello-Interval>10</L1-Hello-Interval>
            <L1-Hello-Multiplier>3</L1-Hello-Multiplier>
            <L2-Hello-Interval>10</L2-Hello-Interval>
            <L2-Hello-Multiplier>3</L2-Hello-Multiplier>
            <ISIS-LSP-Interval>33</ISIS-LSP-Interval>
            <ISIS-L1-CSNP-Interval>10</ISIS-L1-CSNP-Interval>
            <ISIS-L2-CSNP-Interval>10</ISIS-L2-CSNP-Interval>
            <ISIS-L1-PSNP-Interval>10</ISIS-L1-PSNP-Interval>
            <ISIS-L2-PSNP-Interval>10</ISIS-L2-PSNP-Interval>
        </Interface>
    </Interfaces>
    <Routing>
        <ISIS>
            <NET>49.0001.0100.0000.0001.00</NET> <!-- mandatory -->
            <IS-Type>level-1-2</IS-Type>  <!-- level-1, level-2 -->
            <L1-Hello-Interval>10</L1-Hello-Interval>
            <L1-Hello-Multiplier>3</L1-Hello-Multiplier>
            <L2-Hello-Interval>10</L2-Hello-Interval>
            <L2-Hello-Multiplier>3</L2-Hello-Multiplier>
            <LSP-Interval>33</LSP-Interval>
            <LSP-Refresh-Interval>150</LSP-Refresh-Interval>
            <LSP-Max-Lifetime>1200</LSP-Max-Lifetime>
            <L1-LSP-Send-Interval>5</L1-LSP-Send-Interval>
            <L2-LSP-Send-Interval>5</L2-LSP-Send-Interval>
            <L1-LSP-Init-Wait>50</L1-LSP-Init-Wait>
            <L2-LSP-Init-Wait>50</L2-LSP-Init-Wait>
            <L1-CSNP-Interval>10</L1-CSNP-Interval>
            <L2-CSNP-Interval>10</L2-CSNP-Interval>
            <L1-PSNP-Interval>5</L1-PSNP-Interval>
            <L2-PSNP-Interval>5</L2-PSNP-Interval>
            <L1-SPF-Full-Interval>50</L1-SPF-Full-Interval>
            <L2-SPF-Full-Interval>50</L2-SPF-Full-Interval>
```

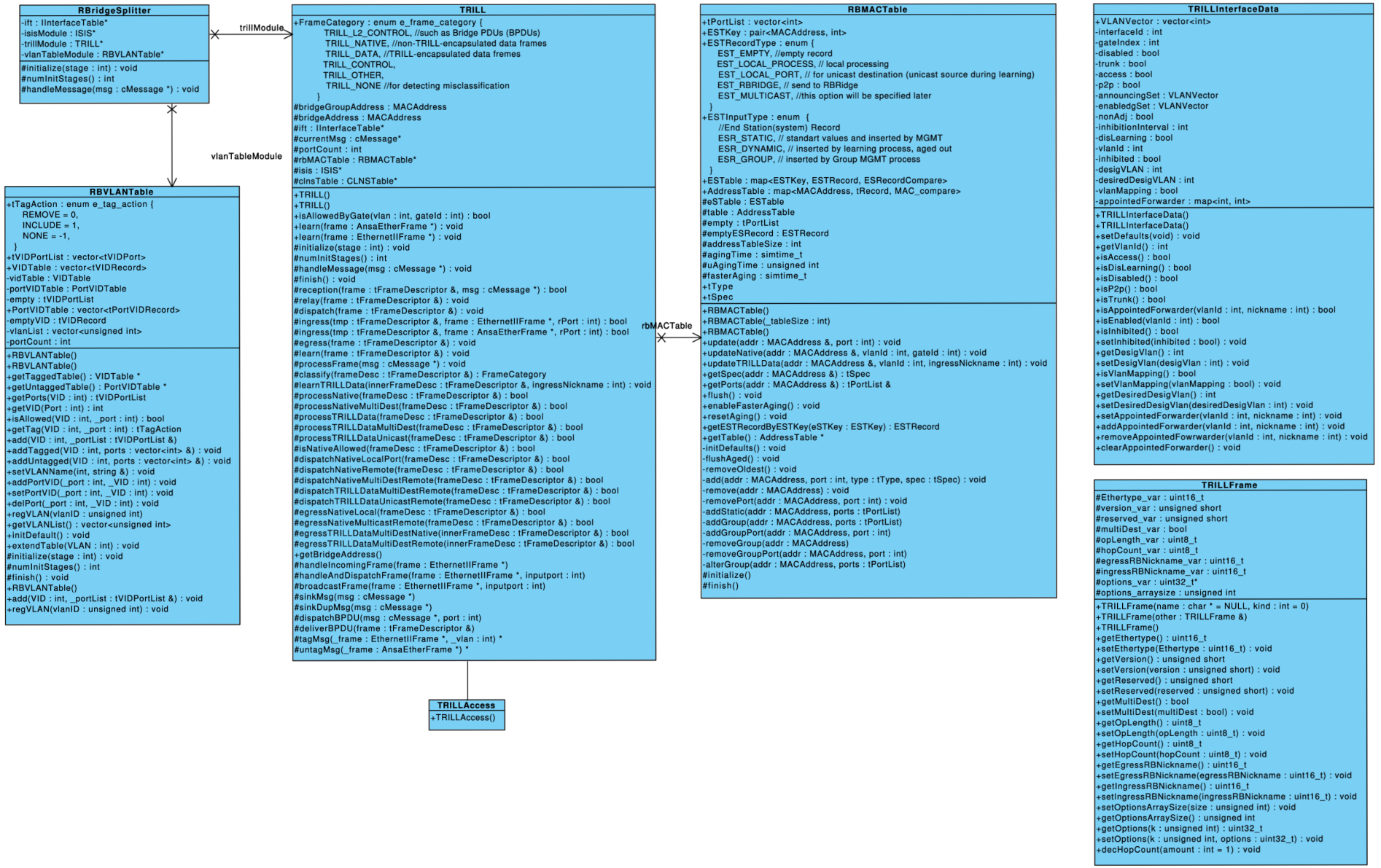
```
41
42          </ISIS>
43      </Routing>
44  </Router>
```

Příloha E

Class diagramy




ISISMessage
#IRPD_var : unsigned char
#length_var : unsigned char
#version_var : unsigned char
#idLength_var : unsigned char
#type_var : unsigned char
#version2_var : unsigned char
#reserved_var : unsigned char
#maxAreas_var : unsigned char
#TLV_var : TLV_t*
#TLV_arraysize : unsigned int
-copy(other : ISISMessage &) : void
#==(ISISMessage &) : bool
+ISISMessage(name : char * = NULL, kind : int = 0)
+ISISMessage(other : ISISMessage &)
+ISISMessage()
+=(other : ISISMessage &) : ISISMessage &
+dup() : ISISMessage *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
+getIRPD() : unsigned char
+setIRPD(IRPD : unsigned char) : void
+getLength() : unsigned char
+setLength(length : unsigned char) : void
+getVersion() : unsigned char
+setVersion(version : unsigned char) : void
+getIdLength() : unsigned char
+setIdLength(idLength : unsigned char) : void
+getType() : unsigned char
+setType(type : unsigned char) : void
+getVersion2() : unsigned char
+setVersion2(version2 : unsigned char) : void
+getReserved() : unsigned char
+setReserved(reserved : unsigned char) : void
+getMaxAreas() : unsigned char
+setMaxAreas(maxAreas : unsigned char) : void
+setTLVArraySize(size : unsigned int) : void
+getTLVArraySize() : unsigned int
+getTLV(k : unsigned int) : TLV_t &
+getTLV(k : unsigned int) : TLV_t &
+setTLV(k : unsigned int, TLV : TLV_t &) : void
<<enumeration>>
TLVtypes
<<Constant>> -TLV_RESERVED = 0
<<Constant>> -AREA_ADDRESS = 1
<<Constant>> -IS_NEIGHBOURS_LSP = 2
<<Constant>> -ES_NEIGHBOURS = 3
<<Constant>> -PDL2_IS = 4
<<Constant>> -PREFIX_NEIGHBOURS = 5
<<Constant>> -IS_NEIGHBOURS_HELLO = 6
<<Constant>> -PADDING = 8
<<Constant>> -LSP_ENTRIES = 9
<<Constant>> -AUTH_INFO_ISO = 10
<<Constant>> -IP_IRI = 128
<<Constant>> -PROTOCOLS_SOPPORTED = 129
<<Constant>> -IP_ERI = 130
<<Constant>> -IRPI = 131
<<Constant>> -IP_ADDRESS = 132
<<Constant>> -AUTH_INFO_RFC = 133
<<Constant>> -TLV_MT_PORT_CAP = 143
<<Constant>> -TLV_MT_PORT_CAP_VLAN_FLAG = 1
<<Constant>> -TLV_MT_PORT_CAP_APP_FWD = 3
<<Constant>> -TLV_TRILL_NEIGHBOR = 145
<<Constant>> -PTP_HELLO_STATE = 240
<<enumeration>>
PDU_Type
<<Constant>> -TRILL_HELLO = 14
<<Constant>> -LAN_L1_HELLO = 15
<<Constant>> -LAN_L2_HELLO = 16
<<Constant>> -PTP_HELLO = 17
<<Constant>> -L1_LSP = 18
<<Constant>> -L2_LSP = 20
<<Constant>> -MTU_PROBE_PDU = 23
<<Constant>> -L1_CSNP = 24
<<Constant>> -L2_CSNP = 25
<<Constant>> -L1_PSNP = 26
<<Constant>> -L2_PSNP = 27
<<Constant>> -MTU_ACK_PDU = 28
<<enumeration>>
CIRCUIT_TYPES
<<Constant>> -RESERVED_TYPE = 0
<<Constant>> -L1_TYPE = 1
<<Constant>> -L2_TYPE = 2
<<Constant>> -L1L2_TYPE = 3
ISISCSNPPacket
#pduLength_var : unsigned short
#sourceID_var : unsigned char[7]
#startLspID_var : unsigned char[8]
#endLspID_var : unsigned char[8]
-copy(other : ISISCSNPPacket &) : void
#==(ISISCSNPPacket &) : bool
+ISISCSNPPacket(name : char * = NULL, kind : int = 0)
+ISISCSNPPacket(other : ISISCSNPPacket &)
+ISISCSNPPacket()
+=(other : ISISCSNPPacket &) : ISISCSNPPacket &
+dup() : ISISCSNPPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
+getPduLength() : unsigned short
+setPduLength(pduLength : unsigned short) : void
+getSourceIDArraySize() : unsigned int
+getSourceID(k : unsigned int) : unsigned char
+setSourceID(k : unsigned int, sourceID : unsigned char) : void
+getStartLspIDArraySize() : unsigned int
+getStartLspID(k : unsigned int) : unsigned char
+setStartLspID(k : unsigned int, startLspID : unsigned char) : void
+getEndLspIDArraySize() : unsigned int
+getEndLspID(k : unsigned int) : unsigned char
+setEndLspID(k : unsigned int, endLspID : unsigned char) : void
ISISCSNPL1Packet
-copy(other : ISISCSNPL1Packet &) : void
#==(ISISCSNPL1Packet &) : bool
+ISISCSNPL1Packet(name : char * = NULL, kind : int = 0)
+ISISCSNPL1Packet(other : ISISCSNPL1Packet &)
+ISISCSNPL1Packet()
+=(other : ISISCSNPL1Packet &) : ISISCSNPL1Packet &
+dup() : ISISCSNPL1Packet *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISCSNPL2Packet
-copy(other : ISISCSNPL2Packet &) : void
#==(ISISCSNPL2Packet &) : bool
+ISISCSNPL2Packet(name : char * = NULL, kind : int = 0)
+ISISCSNPL2Packet(other : ISISCSNPL2Packet &)
+ISISCSNPL2Packet()
+=(other : ISISCSNPL2Packet &) : ISISCSNPL2Packet &
+dup() : ISISCSNPL2Packet *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISMTUPDU
#pduLength_var : uint16_t
#probeID_var : unsigned char[6]
#probeSourceID_var : unsigned char[6]
#ackSourceID_var : unsigned char[6]
-copy(other : ISISMTUPDU &) : void
#==(ISISMTUPDU &) : bool
+ISISMTUPDU(name : char * = NULL, kind : int = 0)
+ISISMTUPDU(other : ISISMTUPDU &)
+ISISMTUPDU()
+=(other : ISISMTUPDU &) : ISISMTUPDU &
+dup() : ISISMTUPDU *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
+getPduLength() : uint16_t
+setPduLength(pduLength : uint16_t) : void
+getProbeIDArraySize() : unsigned int
+getProbeID(k : unsigned int) : unsigned char
+setProbeID(k : unsigned int, probeID : unsigned char) : void
+getProbeSourceIDArraySize() : unsigned int
+getProbeSourceID(k : unsigned int) : unsigned char
+setProbeSourceID(k : unsigned int, probeSourceID : unsigned char) : void
+getAckSourceIDArraySize() : unsigned int
+getAckSourceID(k : unsigned int) : unsigned char
+setAckSourceID(k : unsigned int, ackSourceID : unsigned char) : void
ISISMTUAck
-copy(other : ISISMTUAck &) : void
#==(ISISMTUAck &) : bool
+ISISMTUAck(name : char * = NULL, kind : int = 0)
+ISISMTUAck(other : ISISMTUAck &)
+ISISMTUAck()
+=(other : ISISMTUAck &) : ISISMTUAck &
+dup() : ISISMTUAck *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISMTUProbe
-copy(other : ISISMTUProbe &) : void
#==(ISISMTUProbe &) : bool
+ISISMTUProbe(name : char * = NULL, kind : int = 0)
+ISISMTUProbe(other : ISISMTUProbe &)
+ISISMTUProbe()
+=(other : ISISMTUProbe &) : ISISMTUProbe &
+dup() : ISISMTUProbe *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISLANHelloPacket
#circuitType_var : unsigned char
#sourceID_var : unsigned char[6]
#holdTime_var : unsigned short
#pduLength_var : unsigned short
#priority_var : unsigned char
#lanID_var : unsigned char[7]
-copy(other : ISISLANHelloPacket &) : void
#==(ISISLANHelloPacket &) : bool
+ISISLANHelloPacket(name : char * = NULL, kind : int = 0)
+ISISLANHelloPacket(other : ISISLANHelloPacket &)
+ISISLANHelloPacket()
+=(other : ISISLANHelloPacket &) : ISISLANHelloPacket &
+dup() : ISISLANHelloPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
+getCircuitType() : unsigned char
+setCircuitType(circuitType : unsigned char) : void
+getSourceIDArraySize() : unsigned int
+getSourceID(k : unsigned int) : unsigned char
+setSourceID(k : unsigned int, sourceID : unsigned char) : void
+getHoldTime() : unsigned short
+setHoldTime(holdTime : unsigned short) : void
+getPduLength() : unsigned short
+setPduLength(pduLength : unsigned short) : void
+getPriority() : unsigned char
+setPriority(priority : unsigned char) : void
+getLanIDArraySize() : unsigned int
+getLanID(k : unsigned int) : unsigned char
+setLanID(k : unsigned int, lanID : unsigned char) : void
TRILLHelloPacket
-copy(other : TRILLHelloPacket &) : void
#==(TRILLHelloPacket &) : bool
+TRILLHelloPacket(name : char * = NULL, kind : int = 0)
+TRILLHelloPacket(other : TRILLHelloPacket &)
+TRILLHelloPacket()
+=(other : TRILLHelloPacket &) : TRILLHelloPacket &
+dup() : TRILLHelloPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISL1HelloPacket
-copy(other : ISISL1HelloPacket &) : void
#==(ISISL1HelloPacket &) : bool
+ISISL1HelloPacket(name : char * = NULL, kind : int = 0)
+ISISL1HelloPacket(other : ISISL1HelloPacket &)
+ISISL1HelloPacket()
+=(other : ISISL1HelloPacket &) : ISISL1HelloPacket &
+dup() : ISISL1HelloPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISL2HelloPacket
-copy(other : ISISL2HelloPacket &) : void
#==(ISISL2HelloPacket &) : bool
+ISISL2HelloPacket(name : char * = NULL, kind : int = 0)
+ISISL2HelloPacket(other : ISISL2HelloPacket &)
+ISISL2HelloPacket()
+=(other : ISISL2HelloPacket &) : ISISL2HelloPacket &
+dup() : ISISL2HelloPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISLSPPacket
#pduLength_var : unsigned short
#remLifeTime_var : unsigned short
#lspID_var : unsigned char[8]
#seqNumber_var : unsigned long
#checksum_var : unsigned short
#PATTLSPDBOLIS_var : unsigned char
-copy(other : ISISLSPPacket &) : void
#==(ISISLSPPacket &) : bool
+ISISLSPPacket(name : char * = NULL, kind : int = 0)
+ISISLSPPacket(other : ISISLSPPacket &)
+ISISLSPPacket()
+=(other : ISISLSPPacket &) : ISISLSPPacket &
+dup() : ISISLSPPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
+getPduLength() : unsigned short
+setPduLength(pduLength : unsigned short) : void
+getRemLifeTime() : unsigned short
+setRemLifeTime(remLifeTime : unsigned short) : void
+getLspIDArraySize() : unsigned int
+getLspID(k : unsigned int) : unsigned char
+setLspID(k : unsigned int, lspID : unsigned char) : void
+getSeqNumber() : unsigned long
+setSeqNumber(seqNumber : unsigned long) : void
+getChecksum() : unsigned short
+setChecksum(checksum : unsigned short) : void
+getPATTLSPDBOLIS() : unsigned char
+setPATTLSPDBOLIS(PATTLSPDBOLIS : unsigned char) : void
ISISLSPL1Packet
-copy(other : ISISLSPL1Packet &) : void
#==(ISISLSPL1Packet &) : bool
+ISISLSPL1Packet(name : char * = NULL, kind : int = 0)
+ISISLSPL1Packet(other : ISISLSPL1Packet &)
+ISISLSPL1Packet()
+=(other : ISISLSPL1Packet &) : ISISLSPL1Packet &
+dup() : ISISLSPL1Packet *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISLSPL2Packet
-copy(other : ISISLSPL2Packet &) : void
#==(ISISLSPL2Packet &) : bool
+ISISLSPL2Packet(name : char * = NULL, kind : int = 0)
+ISISLSPL2Packet(other : ISISLSPL2Packet &)
+ISISLSPL2Packet()
+=(other : ISISLSPL2Packet &) : ISISLSPL2Packet &
+dup() : ISISLSPL2Packet *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISPSNPPacket
#pduLength_var : unsigned short
#sourceID_var : unsigned char[7]
-copy(other : ISISPSNPPacket &) : void
#==(ISISPSNPPacket &) : bool
+ISISPSNPPacket(name : char * = NULL, kind : int = 0)
+ISISPSNPPacket(other : ISISPSNPPacket &)
+ISISPSNPPacket()
+=(other : ISISPSNPPacket &) : ISISPSNPPacket &
+dup() : ISISPSNPPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
+getPduLength() : unsigned short
+setPduLength(pduLength : unsigned short) : void
+getSourceIDArraySize() : unsigned int
+getSourceID(k : unsigned int) : unsigned char
+setSourceID(k : unsigned int, sourceID : unsigned char) : void
ISISPSNPL2Packet
-copy(other : ISISPSNPL2Packet &) : void
#==(ISISPSNPL2Packet &) : bool
+ISISPSNPL2Packet(name : char * = NULL, kind : int = 0)
+ISISPSNPL2Packet(other : ISISPSNPL2Packet &)
+ISISPSNPL2Packet()
+=(other : ISISPSNPL2Packet &) : ISISPSNPL2Packet &
+dup() : ISISPSNPL2Packet *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISPSNPL1Packet
-copy(other : ISISPSNPL1Packet &) : void
#==(ISISPSNPL1Packet &) : bool
+ISISPSNPL1Packet(name : char * = NULL, kind : int = 0)
+ISISPSNPL1Packet(other : ISISPSNPL1Packet &)
+ISISPSNPL1Packet()
+=(other : ISISPSNPL1Packet &) : ISISPSNPL1Packet &
+dup() : ISISPSNPL1Packet *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
ISISPTPHelloPacket
#circuitType_var : unsigned char
#sourceID_var : unsigned char[6]
#holdTime_var : unsigned short
#pduLength_var : unsigned short
#localCircuitID_var : unsigned char
-copy(other : ISISPTPHelloPacket &) : void
#==(ISISPTPHelloPacket &) : bool
+ISISPTPHelloPacket(name : char * = NULL, kind : int = 0)
+ISISPTPHelloPacket(other : ISISPTPHelloPacket &)
+ISISPTPHelloPacket()
+=(other : ISISPTPHelloPacket &) : ISISPTPHelloPacket &
+dup() : ISISPTPHelloPacket *
+parsimPack(b : cCommBuffer *) : void
+parsimUnpack(b : cCommBuffer *) : void
+getCircuitType() : unsigned char
+setCircuitType(circuitType : unsigned char) : void
+getSourceIDArraySize() : unsigned int
+getSourceID(k : unsigned int) : unsigned char
+setSourceID(k : unsigned int, sourceID : unsigned char) : void
+getHoldTime() : unsigned short
+setHoldTime(holdTime : unsigned short) : void
+getPduLength() : unsigned short
+setPduLength(pduLength : unsigned short) : void
+getLocalCircuitID() : unsigned char
+setLocalCircuitID(localCircuitID : unsigned char) : void


RBridgeSplitter
-ift : IInterfaceTable*
-isisModule : ISIS*
-trillModule : TRILL*
-vlanTableModule : RBVLANTable*
#initialize(stage : int) : void
#numInitStages() : int
#handleMessage(msg : cMessage *) : void
trillModule
vlanTableModule
TRILL
+FrameCategory : enum e_frame_category {
TRILL_L2_CONTROL, //such as Bridge PDUs (BPDUs)
TRILL_NATIVE, //non-TRILL-encapsulated data frames
TRILL_DATA, //TRILL-encapsulated data fremes
TRILL_CONTROL,
TRILL_OTHER,
TRILL_NONE //for detecting misclassification
}
#bridgeGroupAddress : MACAddress
#bridgeAddress : MACAddress
#ift : IInterfaceTable*
#currentMsg : cMessage*
#portCount : int
#rbMACTable : RBMACTable*
#isis : ISIS*
#clnsTable : CLNSTable*
+TRILL()
+TRILL()
+isAllowedByGate(vlan : int, gateId : int) : bool
+learn(frame : AnsaEtherFrame *) : void
+learn(frame : EthernetIIFrame *) : void
#initialize(stage : int) : void
#numInitStages() : int
#handleMessage(msg : cMessage *) : void
#finish() : void
#reception(frame : tFrameDescriptor &, msg : cMessage *) : bool
#relay(frame : tFrameDescriptor &) : void
#dispatch(frame : tFrameDescriptor &) : void
#ingress(tmp : tFrameDescriptor &, frame : EthernetIIFrame *, rPort : int) : bool
#ingress(tmp : tFrameDescriptor &, frame : AnsaEtherFrame *, rPort : int) : bool
#egress(frame : tFrameDescriptor &) : void
#learn(frame : tFrameDescriptor &) : void
#processFrame(msg : cMessage *) : void
#classify(frameDesc : tFrameDescriptor &) : FrameCategory
#learnTRILLData(innerFrameDesc : tFrameDescriptor &, ingressNickname : int) : void
#processNative(frameDesc : tFrameDescriptor &) : bool
#processNativeMultiDest(frameDesc : tFrameDescriptor &) : bool
#processTRILLData(frameDesc : tFrameDescriptor &) : bool
#processTRILLDataMultiDest(frameDesc : tFrameDescriptor &) : bool
#processTRILLDataUnicast(frameDesc : tFrameDescriptor &) : bool
#isNativeAllowed(frameDesc : tFrameDescriptor &) : bool
#dispatchNativeLocalPort(frameDesc : tFrameDescriptor &) : bool
#dispatchNativeRemote(frameDesc : tFrameDescriptor &) : bool
#dispatchNativeMultiDestRemote(frameDesc : tFrameDescriptor &) : bool
#dispatchTRILLDataMultiDestRemote(frameDesc : tFrameDescriptor &) : bool
#dispatchTRILLDataUnicastRemote(frameDesc : tFrameDescriptor &) : bool
#egressNativeLocal(frameDesc : tFrameDescriptor &) : bool
#egressNativeMulticastRemote(frameDesc : tFrameDescriptor &) : bool
#egressTRILLDataMultiDestNative(innerFrameDesc : tFrameDescriptor &) : bool
#egressTRILLDataMultiDestRemote(innerFrameDesc : tFrameDescriptor &) : bool
+getBridgeAddress()
#handleIncomingFrame(frame : EthernetIIFrame *)
#handleAndDispatchFrame(frame : EthernetIIFrame *, inputport : int)
#broadcastFrame(frame : EthernetIIFrame *, inputport : int)
#sinkMsg(msg : cMessage *)
#sinkDupMsg(msg : cMessage *)
#dispatchBPDU(msg : cMessage *, port : int)
#deliverBPDU(frame : tFrameDescriptor &)
#tagMsg(_frame : EthernetIIFrame *, _vlan : int) *
#untagMsg(_frame : AnsaEtherFrame *) *
RBVLANTable
+tTagAction : enum e_tag_action {
REMOVE = 0,
INCLUDE = 1,
NONE = -1,
}
+tVIDPortList : vector<tVIDPort>
+VIDTable : vector<tVIDRecord>
-vidTable : VIDTable
-portVIDTable : PortVIDTable
-empty : tVIDPortList
+PortVIDTable : vector<tPortVIDRecord>
-emptyVID : tVIDRecord
-vlanList : vector<unsigned int>
-portCount : int
+RBVLANTable()
+RBVLANTable()
+getTaggedTable() : VIDTable *
+getUntaggedTable() : PortVIDTable *
+getPorts(VID : int) : tVIDPortList
+getVID(Port : int) : int
+isAllowed(VID : int, _port : int) : bool
+getTag(VID : int, _port : int) : tTagAction
+add(VID : int, _portList : tVIDPortList &)
+addTagged(VID : int, ports : vector<int> &) : void
+addUntagged(VID : int, ports : vector<int> &) : void
+setVLANName(int, string &) : void
+addPortVID(_port : int, _VID : int) : void
+setPortVID(_port : int, _VID : int) : void
+delPort(_port : int, _VID : int) : void
+regVLAN(vlanID : unsigned int)
+getVLANList() : vector<unsigned int>
+initDefault() : void
+extendTable(VLAN : int) : void
#initialize(stage : int) : void
#numInitStages() : int
#finish() : void
+RBVLANTable()
+add(VID : int, _portList : tVIDPortList &) : void
+regVLAN(vlanID : unsigned int) : void
TRILLAccess
+TRILLAccess()
rbMACTable
RBMACTable
+tPortList : vector<int>
+ESTKey : pair<MACAddress, int>
+ESTRecordType : enum {
EST_EMPTY, //empty record
EST_LOCAL_PROCESS, // local processing
EST_LOCAL_PORT, // for unicast destination (unicast source during learning)
EST_RBRIDGE, // send to RBRidge
EST_MULTICAST, //this option will be specified later
}
+ESTInputType : enum {
//End Station(system) Record
ESR_STATIC, // standart values and inserted by MGMT
ESR_DYNAMIC, // inserted by learning process, aged out
ESR_GROUP, // inserted by Group MGMT process
}
+ESTable : map<ESTKey, ESTRecord, ESRecordCompare>
+AddressTable : map<MACAddress, tRecord, MAC_compare>
#eSTable : ESTable
#table : AddressTable
#empty : tPortList
#emptyESRecord : ESTRecord
#addressTableSize : int
#agingTime : simtime_t
#uAgingTime : unsigned int
#fasterAging : simtime_t
+tType
+tSpec
+RBMACTable()
+RBMACTable(_tableSize : int)
+RBMACTable()
+update(addr : MACAddress &, port : int) : void
+updateNative(addr : MACAddress &, vlanId : int, gateId : int) : void
+updateTRILLData(addr : MACAddress &, vlanId : int, ingressNickname : int) : void
+getSpec(addr : MACAddress &) : tSpec
+getPorts(addr : MACAddress &) : tPortList &
+flush() : void
+enableFasterAging() : void
+resetAging() : void
+getESTRecordByESTKey(eSTKey : ESTKey) : ESTRecord
+getTable() : AddressTable *
-initDefaults() : void
-flushAged() : void
-removeOldest() : void
-add(addr : MACAddress, port : int, type : tType, spec : tSpec) : void
-remove(addr : MACAddress) : void
-removePort(addr : MACAddress, port : int) : void
-addStatic(addr : MACAddress, ports : tPortList)
-addGroup(addr : MACAddress, ports : tPortList)
-addGroupPort(addr : MACAddress, port : int)
-removeGroup(addr : MACAddress)
-removeGroupPort(addr : MACAddress, port : int)
-alterGroup(addr : MACAddress, ports : tPortList)
#initialize()
#finish()
TRILLInterfaceData
+VLANVector : vector<int>
-interfaceId : int
-gateIndex : int
-disabled : bool
-trunk : bool
-access : bool
-p2p : bool
-announcingSet : VLANVector
-enabledgSet : VLANVector
-nonAdj : bool
-inhibitionInterval : int
-disLearning : bool
-vlanId : int
-inhibited : bool
-desigVLAN : int
-desiredDesigVLAN : int
-vlanMapping : bool
-appointedForwarder : map<int, int>
+TRILLInterfaceData()
+TRILLInterfaceData()
+setDefaults(void) : void
+getVlanId() : int
+isAccess() : bool
+isDisLearning() : bool
+isDisabled() : bool
+isP2p() : bool
+isTrunk() : bool
+isAppointedForwarder(vlanId : int, nickname : int) : bool
+isEnabled(vlanId : int) : bool
+isInhibited() : bool
+setInhibited(inhibited : bool) : void
+getDesigVlan() : int
+setDesigVlan(desigVlan : int) : void
+isVlanMapping() : bool
+setVlanMapping(vlanMapping : bool) : void
+getDesiredDesigVlan() : int
+setDesiredDesigVlan(desiredDesigVlan : int) : void
+setAppointedForwarder(vlanId : int, nickname : int) : void
+addAppointedForwarder(vlanId : int, nickname : int) : void
+removeAppointedFowrwarder(vlanId : int, nickname : int) : void
+clearAppointedForwarder() : void
TRILLFrame
#Ethertype_var : uint16_t
#version_var : unsigned short
#reserved_var : unsigned short
#multiDest_var : bool
#opLength_var : uint8_t
#hopCount_var : uint8_t
#egressRBNickname_var : uint16_t
#ingressRBNickname_var : uint16_t
#options_var : uint32_t*
#options_arraysize : unsigned int
+TRILLFrame(name : char * = NULL, kind : int = 0)
+TRILLFrame(other : TRILLFrame &)
+TRILLFrame()
+getEthertype() : uint16_t
+setEthertype(Ethertype : uint16_t) : void
+getVersion() : unsigned short
+setVersion(version : unsigned short) : void
+getReserved() : unsigned short
+setReserved(reserved : unsigned short) : void
+getMultiDest() : bool
+setMultiDest(multiDest : bool) : void
+getOpLength() : uint8_t
+setOpLength(opLength : uint8_t) : void
+getHopCount() : uint8_t
+setHopCount(hopCount : uint8_t) : void
+getEgressRBNickname() : uint16_t
+setEgressRBNickname(egressRBNickname : uint16_t) : void
+getIngressRBNickname() : uint16_t
+setIngressRBNickname(ingressRBNickname : uint16_t) : void
+setOptionsArraySize(size : unsigned int) : void
+getOptionsArraySize() : unsigned int
+getOptions(k : unsigned int) : uint32_t
+setOptions(k : unsigned int, options : uint32_t) : void
+decHopCount(amount : int = 1) : void