VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

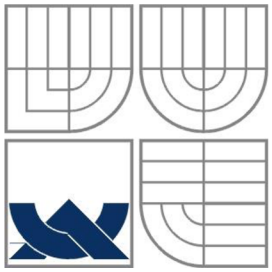
MODELOVÁNÍ SMĚROVACÍHO PROTOKOLU BABEL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

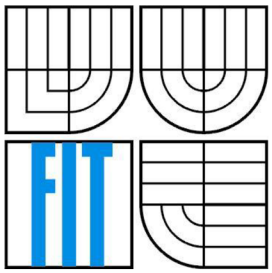
AUTOR PRÁCE
AUTHOR

Bc. VÍT REK

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ SMĚROVACÍHO PROTOKOLU BABEL

MODELLING OF BABEL ROUTING PROTOCOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VÍT REK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR VESELÝ

BRNO 2015

Abstrakt

Tato práce se zabývá simulováním směrovacího protokolu Babel. Cílem je vytvoření implementace simulačního modelu pro prostředí OMNeT++. Součástí textu práce je popis tohoto protokolu a základních principů simulace počítačových sítí v prostředí OMNeT++ s použitím knihovny INET. Dále jsou v práci diskutovány již existující implementace a je předložen návrh simulačního modelu, následovaný popisem jeho implementace. Nakonec je provedeno ověření správnosti vytvořeného modelu.

Abstract

This thesis deals with the simulation of a Babel routing protocol. The goal is to create implementation of simulation model for OMNeT++ simulator. The text includes a description of the protocol and basic principles of computer network simulation in OMNeT++ environment using an INET library. Furthermore, the text discussed existing implementations and submits a proposal of a simulation model, followed by description of its implementation. Finally, the correctness of created model is verified.

Klíčová slova

Simulace sítí, směrovací protokol, Babel, OMNeT++, INET, ANSA.

Keywords

Network Simulation, routing protocol, Babel, OMNeT++, INET, ANSA.

Citace

Rek Vít: Modelování směrovacího protokolu Babel, diplomová práce, Brno, FIT VUT v Brně, 2015

Modelování směrovacího protokolu Babel

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Veselého.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Vít Rek
27. května 2015

Poděkování

Mé poděkování patří Ing. Vladimíru Veselému za trpělivost, odbornou pomoc, poskytnuté rady a připomínky při psaní této práce. Na oplátku připojuji jednoduchý, ale výborný recept.

Vepřové plátky na houbách
Vepřové plátky lehce naklepeme, osolíme a opepříme. Prudce je opečeme po obou stranách, tak aby se maso zatáhlo, a dočasně vyjmeme. Osmažíme nakrájenou cibuli do růžova, ke které vrátíme maso zpět. Podlijeme vodou, přidáme sušené houby (nejlépe žampiony) a s občasným podléváním dusíme do měkka. Poté zahustíme hladkou moukou rozmíchanou ve vodě a dodusíme. Podáváme s rýží.

Také děkuji mé rodině za důvěru a nezměrnou podporu.

Obsah

1 Úvod

V průběhu posledních několika desetiletí prošla celosvětová počítačová síť Internet velkým vývojem. Internet se značně rozrostl a stalo se z něj komunikační médium, bez kterého si život v dnešní době dokážeme představit jen velice obtížně. Neustálým rozšiřováním jsou na jím používané technologie kladeny stále nové požadavky. Pro uspokojení těchto požadavků jsou vyvíjeny nové mechanismy, či stávající zdokonalovány. Tento vývoj však bývá zpravidla velmi náročný, což jej činí zdlouhavým a drahým. Z těchto důvodů jsou předmětem výzkumu způsoby jeho zjednodušení. A právě jedním takovýmto způsobem je počítačová simulace. Ta umožňuje provádění experimentů s vyvíjenými systémy, a to použitím výpočetní techniky, bez nutnosti výroby prototypů. Pro simulaci počítačových sítí je výzkumníky hojně využíváno prostředí OMNeT++, rozšířené knihovnou INET.

Tato práce se zabývá směrovacím protokolem Babel. Jejím hlavním cílem je rozšířit simulační prostředí OMNeT++ o jeho podporu, a tím přispět ke zjednodušení vývoje a testování daného protokolu. Ve druhé kapitole je popsána jeho koncepce, chování a vlastnosti. Taktéž je nastíněno základní porovnání s jinými směrovacími protokoly, a rozebrána vhodnost jeho nasazení v určitých prostředích. Následuje výčet existujících implementací. Je diskutován jejich aktuální stav, podpora síťových prostředí a způsob konfigurace. Třetí kapitola se věnuje popisu základních principů simulace počítačových sítí v prostředí OMNeT++. Je uveden princip diskrétní simulace, popsána hierarchie simulačních modelů a představeny knihovny INET a ANSAINET. Součástí je i popis rozšíření simulačního modelu směrovacího protokolu EIGRP o podporu IPv6, které bylo v rámci této práce vytvořeno. Ve čtvrté kapitole je předložen návrh implementace simulačního modelu směrovacího protokolu Babel. Je navržena struktura složeného modulu, abstraktní datové typy a způsob konfigurace směrovače. Samotná implementace a v ní použitá řešení jsou detailně rozvedená v kapitole páté. Ta se věnuje způsobu práce s reálným časem v simulačním prostředí, konceptu umožňujícím shlukování a kompresi přenášených zpráv a použitým datovým strukturám. Obsah šesté kapitoly pak porovnává simulační model s reálnou implementací a diskutuje získané výstupy.

2 Směrovací protokol Babel

Tato kapitola popisuje směrovací protokol *Babel*. Vychází z dokumentu RFC 6126 [1], ovšem snaží se obsáhnout i změny a rozšíření toho protokolu, dané dalším vývojem a výzkumem v této oblasti. Na začátku kapitoly jsou popsány vlastnosti tohoto protokolu a ten je zařazen do odpovídajících skupin podle obecného dělení směrovacích protokolů. Dále jsou uvedeny problémové situace v síti a nastíněny způsoby jejich řešení. Následuje popis funkcionality protokolu a datových struktur. V závěru kapitoly je uveden výčet existujících implementací.

Babel je dynamický směrovací protokol určený pro *paketově přepínané* (*packet-switched*) počítačové sítě, popsaný dokumentem RFC 6126 [1]. Hlavním cílem směrovacích protokolů je určování optimálních cest do cíle, a to dynamicky v reakci na změny topologie a stavu sítě. Babel se řadí do skupiny protokolů pracujících na základě *délky vektoru* (*Distance-vector*). Na rozdíl od skupiny pracující se *stavem linky* (*Link-state*), neznají tyto protokoly celou topologii sítě, ale pouze její část. Informace o topologii jsou sdíleny sousedními směrovači. V jiném rozdělení směrovacích protokolů, podle okamžiku výpočtu optimální cesty, náleží Babel do skupiny *proaktivních*. Proaktivní protokoly aktivně vytvářejí a udržují směrovací cesty, ať již datový provoz využívající dané cesty existuje, či nikoliv. Opakem proaktivních protokolů jsou protokoly *reaktivní*, vytvářející cesty až v okamžiku existence požadavku přenosu datového provozu. Při výpočtu optimálních cest používá Babel distribuovanou variantu *Bellmanova-Fordova algoritmu* [11], doplněnou o různá vylepšení pro zdokonalení jeho vlastností. Až na určitou vyjímku, je protokolem Babel garantována topologie bez výskytu *směrovacích smyček*. Zmíněnou vyjímkou je propagace stejné cílové sítě více různými směrovači. V takovémto případě může dojít k výskytu dočasné smyčky, která však po čase úměrném její velikosti, zmizí.

První práce na protokolu Babel započaly v roce 2007. Jeho autorem je *Juliusz Chroboczek* z Université Paris-Diderot. Původně byl tento protokol koncipován pro použití v bezdrátových *mesh* sítích. Avšak jeho návrh umožňuje použití různých způsobů určování cen linek, výpočtů metrik a politik výběru cest, a navíc je natolik flexibilní, že je jeho nasazení vhodné i v jiných prostředích. Například v bezdrátových sítích s použitím způsobu určení ceny linky *ETX* (*Estimated Transmission Cost*, [25]), v drátových sítích s použitím strategie *2-out-of-3* a ve virtuálních sítích využívajících již existující sítě s použitím strategie zohledňující zpoždění. Používané strategie jsou podrobně popsány v kapitole 2.4.6 Také je hojně využíván pří výzkumu a experimentování v oblasti počítačových sítí. Příklady prací jsou např. směrování zohledňující rušení mezi rádiovými kanály (*Diversity routing*, [26]), směrování zohledňující zdrojové adresy (*Source-specific routing*, [28]), směrování podle zpoždění (*Delay-based routing*, [27]) a další.

Babel čerpá inspiraci a přebírá některé mechanismy používané u jiných směrovacích protokolů. Například problém *počítání do nekonečna* (*Counting-to-infinity,* kapitola 2.3.2), trpějícím obecně

všechny směrovací protokoly pracující na základě délky vektoru, je řešen stejným způsobem jako u protokolu *EIGRP* [24], pomocí *podmínky vhodnosti* (*feasibility condition, FC*), konkrétně typem *SNC* (Source Node Condition, [10]). Při použití této podmínky může ovšem docházet k hladovění. Pro jeho eliminaci je na rozdíl od EIGRP použit mechanismus *sekvenčních čísel*, do značné míry inspirovaný z protokolu *DSDV* [22]. Sekvenční číslo určuje čerstvost informace o cestě. Čerstvější informace jsou upřednostňovány. Oproti protokolu DSDV nejsou sekvenční čísla navyšována periodicky, ale pouze jako reakce na přijetí požadavku na zvýšení sekvenčního čísla. Tím je celý proces urychlen a předchází se tak dočasnému hladovění, ke kterému by docházelo při čekání na periodické navýšení sekvenčního čísla. Problematika hladovění je detailně rozvedena v kapitole 2.3.4.

Babel podporuje protokoly *IPv4* (Internet Protocol verze 4) i *IPv6* (Internet Protocol verze 6), a to nezávisle na aktuálně zprávami použitém protokolu síťové vrstvy. Pro přenos zpráv používá protokol *UDP* (User Datagram Protocol). Nespolehlivost přenosu řeší v určitých případech vícenásobným odesláním stejné zprávy. Při zasílání zpráv se snaží být velmi efektivní. Síťové adresy, *prefixy sítí* a identifikátory směrovačů jsou komprimovány a duplicitní výskyty informací jsou odstraňovány. Zasílání informací je záměrně pozdržováno tak, aby přenášené zprávy obsahovaly více informací, a tím se minimalizovala režie nižších vrstev při přenosu. Pokud je to možné, bývají zprávy zasílány skupinově (multicast), což přináší snížení počtu odeslaných zpráv a také nedochází k procesu překladu *adresy síťové vrstvy* na *adresu linkové vrstvy* (procesy *ARP* a *Neighbour Discovery*). Zprávy *Update*, nesoucí informace o topologii, jsou odesílány jak periodicky, tak i neplánovaně jako reakce na změnu v síti. Dalšími periodicky odesílanými zprávami jsou zprávy *Hello*, sloužící k detekci funkčnosti spojení mezi sousedními směrovači. Odpověďmi na zprávy Hello jsou zprávy *IHU* (“*I Heard You*“), které však mohou být zasílané méně často něž jsou Hello zprávy doručovány. Zprávy Hello společně se zprávami IHU slouží pro zjišťování kvality linky mezi danými sousedy. Rozšíření přenášených směrovacích informací o zabezpečení pomocí *autentizace*, využívající kryptografické principy, je popsáno v dokumentu RFC 7298 [4].

2.1 Vlastnosti protokolu Babel

Hlavní vlastností, která činí protokol Babel vhodným pro nasazení v nestabilních sítích, je jeho důrazné omezování četnosti a délky nežádoucích stavů, jako jsou *směrovací smyčky* a *černé díry*, během konvergence sítě. Po změně v síti obvykle zůstává topologie bezsmyčková. V reakci na tuto změnu Babel zkonverguje síť do stavu zajišťující bezsmyčkové a souvislé propojení sítě, které však nemusí být nutně optimální. Ve většině případů se tento proces zcela obejde bez výměny zpráv mezi směrovači. Poté, v řádu minut, topologie zkonverguje do optimálního stavu.

Pro úplnost, na tomto místě definujme některé dále v textu používané pojmy: *Cesta* z uzlu v_0 do v_k je posloupnost uzlů začínající uzlem v_0 a končící v_k, kde sousední prvky této posloupnosti jsou vzájemně propojené hranou, a ve které se uzly neopakují. To lze zapsat formálně, jako:

$$\langle \boldsymbol{v_0}, \boldsymbol{v_1}, \boldsymbol{v_2}, \dots \boldsymbol{v_k} \rangle, kde$$
$$v_m \neq v_n \mid m \neq n \; pro \; m, n = 0,1, \dots k,$$
$$(v_{i-1}, v_i) \in E \; pro \; i = 1,2, \dots k. \tag{2.1}$$

E značí množinu hran, *V* množinu uzlů, *v* uzel, (v_r, v_s) hranu z uzlu v_r do v_s. *Délka cesty* je počet hran dané cesty. *Velikost sítě* je určena délkou nejdelší cesty v dané síti.

Směrovací smyčka je posloupnost uzlů začínající uzlem v_0 a končící v_k takovým, že $v_k = v_0$ kde sousední prvky této posloupnosti jsou vzájemně propojené hranou, a ve které se uzly, s vyjímkou uzlu v_0, neopakují. Zapsáno jako:

$$\langle \boldsymbol{v_0}, \boldsymbol{v_1}, \boldsymbol{v_2}, \dots \boldsymbol{v_k} \rangle, kde$$
$$v_0 = v_k,$$
$$v_m \neq v_n \mid m \neq n \; pro \; m, n = 0,1, \dots k-1,$$
$$(v_{i-1}, v_i) \in E \; pro \; i = 1,2, \dots k. \tag{2.2}$$

Velikost smyčky je definována jako počet hran dané směrovací smyčky. *Bezsmyčková topologie* je topologie, ve které se nevyskytuje žádná směrovací smyčka. Pojem *černá díra* označuje směrovač, který z důvodu neznalosti cesty do cíle daný datový provoz likviduje.

Nejvýznamnějšími vlastnostmi protokolu Babel jsou:

- Pro každý prefix, který je propagován nanejvýš jedním směrovačem, je vždy garantováno bezsmyčkové prostředí.
- Pro prefix propagovaný více než jedním směrovačem, může dojít k vytvoření dočasné směrovací smyčky. Tato smyčka je odstraněna během doby úměrné její velikosti. Směrovače v ní zahrnuté ji v budoucnosti (omezené časem úklidu *GC*, *Garbage Collection*) již nevytvoří.
- Libovolná černá díra je odstraněna během doby úměrné *velikosti sítě* (za předpokladu přiměřené ztráty paketů v síti).

Směrovače úspěšně naváží vztah sousedství i v případě, že používají rozdílné parametry časovačů. To je výhodné v bezdrátových sítích, kde se často vyskytují mobilní prvky s důrazem na úsporu energie.

2.2 Omezení

Babel má dvě významná omezení, která jej činí nevhodným pro nasazení v určitých prostředích. Zaprvé, namísto použití spolehlivého přenosu zpráv využívá pravidelné zasílání směrovacích informací. To ve velkých stabilních sítích přináší generování většího množství datového provozu, než je tomu u protokolů zasílajících informace pouze při změně topologie. Nasazení těchto protokolů bude

v daném prostředí vhodnější, než použití protokolu Babel. Zadruhé, Babel není vhodný v mobilních sítích s automatickou agregací prefixů. A to díky použití *doby prodlevy* (*Hold time*, 2.3.6), zajišťující ochranu před vznikem smyčky mezi různě dlouhými prefixy. Ve výsledku, pokud se deagregovaný prefix stane agregovaným, bude po dobu prodlevy nedostupný.

2.3 Koncepce protokolu

Babel je protokol pracující na základě délky vektoru: staví na *Bellmanově-Fordově algoritmu* [11], který doplňuje mnoha vylepšeními, buď zcela zabraňujícími vytvoření smyčky, nebo zajišťujícími jejich včasné odstranění. Přesněji, je použita distribuovaná verze tohoto algoritmu, ve které probíhá výpočet pro všechny zdroje směrovacích informací (směrovače propagující cílovou síť) paralelně. Babel ohodnocuje linky mezi dvěmi směrovači. Toto hodnocení se nazývá *cena linky*, a formálně se jedná o abstraktní hodnotu přidělenou hraně mezi dvěmi uzly grafu. Cenu linky mezi uzly A a B označujeme jako *C(A, B)*. *Metrikou cesty* je označována suma cen hran, které jsou využity při průchodu cestou. Cílem směrovacích protokolů je vypočítat pro každý uzel S strom cest s nejnižšími metrikami do uzlu S. Protokol Babel určuje cenu linky z periodicky zasílaných zpráv *Hello* a odpovědí na tyto zprávy (zprávy *IHU*). Způsoby jakými to provádí jsou uvedeny v kapitole 2.4.6.

2.3.1 Bellmanův-Fordův algoritmus

Jedná se o algoritmus pro hledání nejkratších cest v orientovaném grafu. Je schopný pracovat s libovolným ohodnocením hran, a to i záporným. Označíme-li množinu uzlů *V* a množinu hran *E*, pak lze Ballmanův-Fordův algoritmus zapsat pseudokódem Pseudokód 2.1. Každý uzel si pro cíl S udržuje dvě informace. První informací je *odhad vzdálenosti* do cíle S, označený *jako D[A]*. Hodnota D[A] je vždy horním odhadem délky nejkratší cesty do S. Při běhu algoritmu se tento odhad zpřesňuje. Druhou informací je *zvolený následník*, (sousední směrovač, přes který vede cesta do cíle, tzv. next-hop či successor), označován jako *NH[A]*.

```
BELLMAN-FORD((V,E),C,S)
 1 for každý uzel v ∈V do
 2       D[v] ← ∞
 3       NH[v] ← NIL
 4 D[S] ← 0
 5 for i ← 1 to n-1 do
 6       for každou hranu (u,v)∈E do
 7             if D[v] > D[u] + C(u,v) then
 8                   D[v] ← D[u] + C(u,v)
 9                   NH[v] ← u
10 for každou hranu (u,v)∈E do
11       if D[v] > D[u] + C(u,v) then
12             return FALSE
13 return TRUE
```

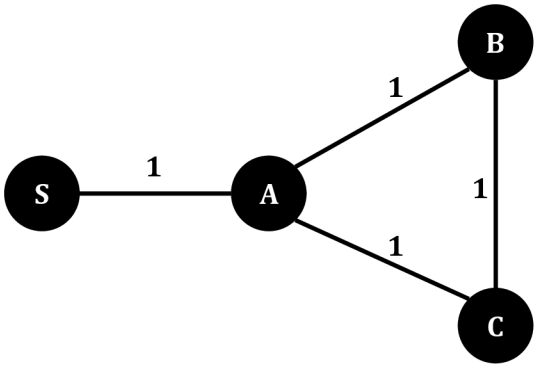
Pseudokód 2.1: Bellmanův-Fordův algoritmus

Z pohledu samotného uzlu pak výpočet pomocí distribuované varianty Bellmanova-Fordova algoritmu probíhá takto: Na začátku výpočtu nastavíme D[A] na nekonečno, D[S] na nulu, NH[A] je nedefinován. Každý uzel B periodicky posílá všem jeho sousedním uzlům informace o cestách, obsahující D[B]. Při přijetí této zprávy uzel A nejprve ověří, zda je soused B zvoleným následníkem pro daný cíl. Jedná-li se o tento případ, jde o aktualizaci již zvolené cesty a D[A] je jednoduše nastaveno na *C(A, B) + D[B]*. V opačném případě je porovnána hodnota *C(A, B) + D[B]* s aktuální hodnotou D[A]. Je-li hodnota *C(A, B) + D[B]* nižší znamená to, že zpráva oznamuje lepší cestu než je aktuálně zvolena. Proto je NH[A] nastaveno na B, a D[A] nastaveno na *C(A, B) + D[B]*.

V protokolu Babel je Bellmanův-Fordův algoritmus doplněn o určitá vylepšení. Jedním z nich je zasílání i neplánovaných aktualizačních informací, jako okamžitá reakce na změnu v síti. Díky tomu je urychlen proces konvergence sítě. Dalším vylepšením je udržování alternativních cest, vedoucích přes jiné než aktuálně zvolené následníky, v paměti. Při výpadku aktuálně zvolené cesty mohou být tyto alternativy ihned použity.

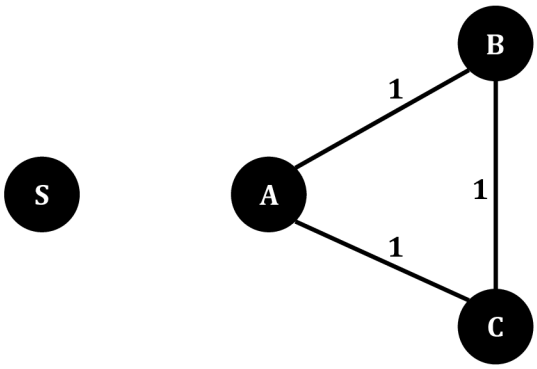
2.3.2 Problém počítání do nekonečna

Je dobře známo, že naivní použití Bellmanova-Fordova algoritmu pro distribuované směrování, může po změně topologie způsobit výskyt dočasné smyčky. Tento problém je demonstrován následujícím příkladem. Mějme topologii znázorněnou na obrázku Obrázek 2.1:



Obrázek 2.1: Topologie bez výpadku

Po zkonvergování platí, že *D[B] = D[C] = 2 a NH[B] = NH[C] = A*. Nyní předpokládejme výpadek linky mezi S a A, jak zachycuje Obrázek 2.2:



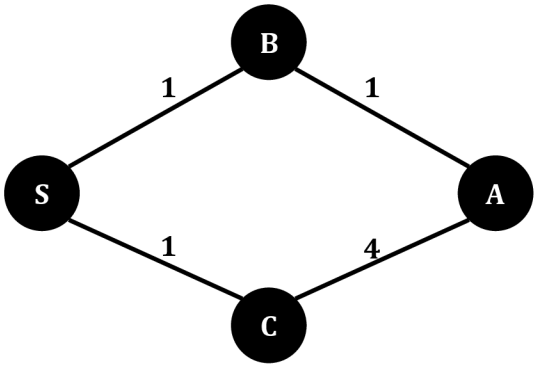
Obrázek 2.2: Topologie s nefunkční linkou mezi uzly S a A

Po detekci výpadku linky změní směrovač A svého zvoleného následníka pro cíl S na směrovač B. To protože směrovač B nemá informaci o výpadku této linky a nadále oznamuje dostupnost cíle s metrikou *D[B] = 2*, což je z pohledu směrovače A jiná funkční cesta do cíle S. Metriku této cesty určí jako *D[A] = C(A, B) + D[B] = 1 + 2 = 3*, a začne ji oznamovat sousedním uzlům. Toto oznámení obdrží směrovač B, a jelikož je A jeho zvoleným následníkem pro cíl S, hodnotu nijak neporovnává a přímo nastavuje hodnotu *D[B] = C(A, B) + D[A] = 1 + 3 = 4*. Takto se proces opakuje a metrika je průběžně navyšována až do doby dosažení „nekonečna“, které je definováno jako dostatečně velká hodnota metriky. Dosažením „*nekonečna*“ je cesta anulována, ovšem během tohoto procesu byla v síti vytvořena směrovací smyčka.

2.3.3 Podmínka vhodnosti

Pro řešení problému počítání do nekonečna Babel používá tzv. podmínku vhodnosti (Feasibility Condition). Podmínka vhodnosti je podmínka kladená příchozím oznámením o cestách. Ignorování oznámení nesplňující tuto podmínku všemi směrovači v síti garantuje, že v topologii nedojde k vytvoření smyčky. Existuje mnoho různých podmínek vhodnosti. Například podmínka vhodnosti použitá ve směrovacích protokolech *DSDV* (Destination-Sequenced Distance-Vector, [22]) a *AODV* (Ad hoc On-Demand Distance Vector, [23]), dále v textu označovaná jako *DSDV-FC*, vychází z následujícího pozorování: směrovací smyčka může vzniknout pouze poté, co směrovač změnil zvolenou cestu do cíle na cestu s větší metrikou, než byla u té předešlé. Tudíž cesta může být považována za vhodnou, pouze pokud její metrika z lokálního pohledu směrovače, není větší než metrika aktuálně zvolené cesty. To znamená, že příchozí oznámení nesoucí metriku D[B] je přijato směrovačem A pouze pokud je splněna podmínka $\mathrm{C(A,B)} + \mathrm{D[B]} \leq \mathrm{D[A]}$. Dodržují-li toto omezení všechny směrovače, je metrika na každém směrovači nerostoucí, a následující invariant je vždy splněn: má-li směrovač A zvoleného za následníka B, pak $\mathrm{D[B]} < \mathrm{D[A]}$, což implikuje, že výsledný graf neobsahuje smyčky.

Babel však používá mírně odlišnou podmínku vhodnosti. Nazývá se SNC (Source Node Condition, [10]) a je používána směrovacím protokolem EIGRP. Definuje vhodnou vzdálenost (Feasibility Distance), označovanou *FD[A]*, jako nejmenší metriku do cíle S, kterou kdy směrovač A oznamoval svým sousedům. Oznámení přijaté směrovačem A od B splňuje FC, pokud je metrika D[B] striktně menší než FD[A], tj. $D[B] < FD[A]$. SNC je méně přísná než DSDV-FC, což lze jednoduše demonstrovat. Uvažujme uzel A dodržující DSDV-FC, pak je D[A] nerostoucí, tudíž vždy platí $D[A] <= FD[A]$. Při přijmu oznámení od uzlu B s metrikou D[B] vyhovující DSDV-FC platí podmínka $C(A, B) + D[B] <= D[A]$, tudíž $D[B] < D[A]$. A jelikož $D[A] <= FD[A]$, pak platí $D[B] < FD[A]$. Jak se tento rozdíl projeví, je demonstrováno na topologii znázorněné obrázkem Obrázek 2.3:



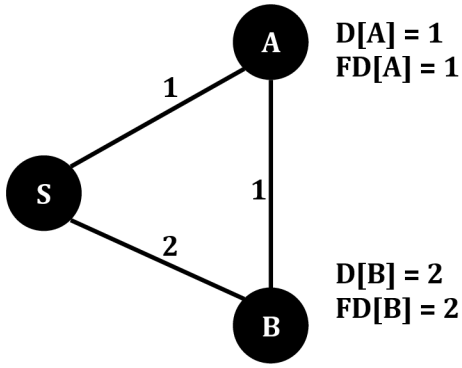
Obrázek 2.3: Topologie pro demonstraci FC

Uzel A zvolil cestu do cíle S přes B s parametry $D[A] = FD[A] = 2$. Jelikož $D[C] = 1 < FD[A]$, alternativní cesta přes uzel C je vhodná, a to i přes to, že její metrika $C(A, C) + D[C] = 5$ je vyšší než aktuálně zvolená cesta.

K důkazu, že méně přísná podmínka SNC stále zaručuje bezsmyčkovost připomeňme, že v době příjmu oznámení uzlem A od B metrika D[B] není menší než FD[B], je však menší než FD[A], a tudíž platí $FD[B] < FD[A]$. Jelikož tato vlastnost platí i při odesílání oznámení uzlem A, zůstává platnou po celou dobu, což zajišťuje bezsmyčkovost výsledné topologie. Podrobný důkaz podmínky SNC je možné nalézt ve článku [10].

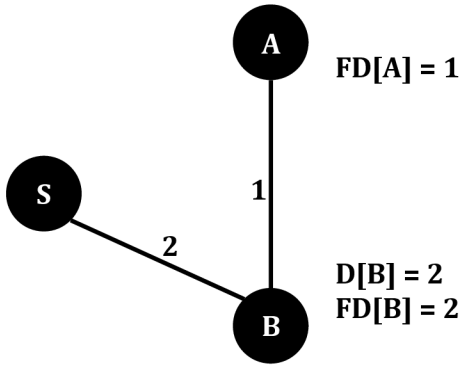
2.3.4 Problém hladovění

Použití podmínky vhodnosti může způsobovat problém hladovění. K tomu dochází při vyčerpání všech vhodných cest do cíle. Existující dostupná cesta není použita, protože nesplňuje FC. Tento jev lze demonstrovat na topologii Obrázek 2.4:



Obrázek 2.4: Topologie pro demonstraci problému hladovění

Oba uzly, A i B, zvolily do cíle S přímou linku, tedy (S, A), respektive (S, B). Nyní předpokládejme selhání linky (S, A):

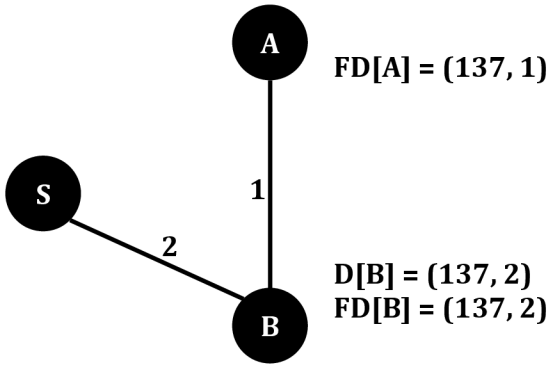


Obrázek 2.5: Topologie s nefunkční linkou mezi uzly S a A, uzel A trpí hladověním

Jediná dostupná cesta z A do S, vedoucí přes uzel B, nesplňuje FC, a proto není použita. Díky tomu uzel A trpí hladověním. Ve směrovacím protokolu EIGRP, odkud je FC převzata, je tento jev řešen globální synchronizací všech uzlů s cílovým uzlem. Tento proces synchronizace všech uzlů je též označován jako „*aktivní fáze*“. Oproti tomu protokol Babel reaguje na hladovění méně drasticky. Využívá k tomu mechanismus sekvenčních čísel, používaný i protokoly DSDV a AODV. Ten doplňuje oznamovanou informaci o cestě o neklesající číslo, které je propagováno napříč sítí beze změny. Měnit, a to pouze zvyšovat, jej může jedině samotný zdroj informace o cílové síti. Dvojice *(s, m)*, kde *s* značí sekvenční číslo a *m* metriku, je nazývána *vzdálenost*. Sekvenční číslo je do podmínky vhodnosti

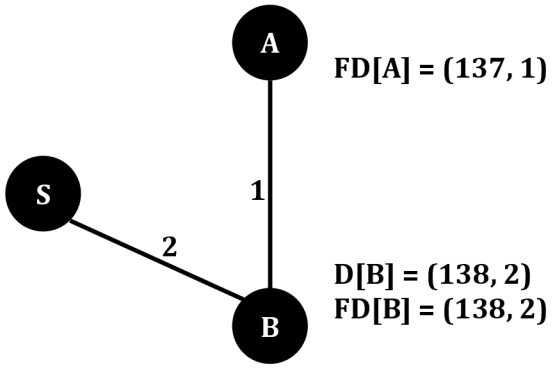
zakomponováno tak, aby zaručovalo, že bude upřednostněna čerstvější informace, a stále garantovala bezsmyčkovost topologie. Porovnání vzdáleností je definováno následovně:

Je-li $FD[A] = (s', m')$, pak vzdálenost (s, m) splňuje podmínku FC, platí-li $s > s' OR$ $(s = s' AND\ m < m')$. Jak se sekvenční čísla projeví v praxi, je demonstrováno na topologii shodné s minulým příkladem, která předpokládá, že sekvenční číslo uzlu S je aktuálně *137*, viz Obrázek 2.6:



Obrázek 2.6: Stav sekvenčních čísel při výpadku

Po navýšení sekvenčního čísla uzlem S, a jeho propagací topologií až k uzlu B, dostáváme situaci znázorněnou obrázkem Obrázek 2.7:



Obrázek 2.7: Stav sekvenčních čísel po doručení aktualizace uzlu B

V tomto stavu vzdálenost oznamovaná uzlem B splňuje FC a A může použít cestu vedoucí přes B. Problém hladovění byl tímto odstraněn.

V předchozím textu je popsán funkční mechanismus obrany proti hladovění, ovšem není uveden podnět vyvolávající zvýšení sekvenčního čísla. To může být prováděno periodicky. Tak je tomu například v protokolu DSDV. Při použití tohoto přístupu nemá směrovač žádný prostředek, jak by navýšení sekvenčního čísla vyvolal. Proto musí na periodické navýšení čekat. Během čekání je cíl pro daný uzel nedostupný – dochází k dočasnému hladovění. Navíc, v případě stabilních cest, dochází k většině navyšování zcela zbytečně. Z těchto důvodů používá Babel jiný přístup. Když uzel detekuje

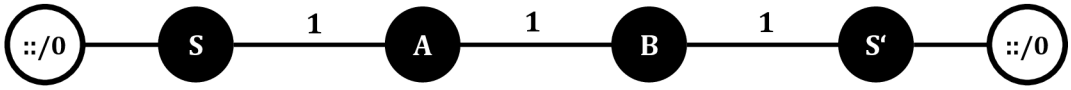
možnost hladovění, explicitně požádá o navýšení sekvenčního čísla. Tento požadavek je přeposílán jednotlivými uzly až k uzlu propagující cílovou síť, a to bez ohledu na podmínku vhodnosti. Při přijetí požadavku, zvýší zdrojový uzel sekvenční číslo a aktualizaci odešle všesměrově do sítě. Ta je po čase doručena i uzlu žádajícího o navýšení. Ne všechny požadavky budou doručeny ke zdroji informace, jelikož jsou posílány i po linkách, které nejsou aktuálně funkční. Jelikož jsou požadavky přeposílány bez ohledu na FC, může dojít k výskytu smyčky. Z tohoto důvodu musí uzly duplicitní požadavky ignorovat. Navíc je přeposílání zprávy požadavku omezeno počtem přeskoků.

2.3.5 Vícenásobná propagace prefixu sítě

Dosud byl v textu uvažován pouze případ, kdy je každá cílová síť propagována nanejvýš jedním směrovačem. V praxi je však často nezbytné propagovat danou síť více směrovači. Například výchozí cesta bývá propagována všemi hraničnímu směrovači směrovací domény.

Jelikož synchronizace sekvenčních čísel mezi uzly je problematická, Babel považuje cesty do cílové sítě propagované různými směrovači za odlišné. Každé oznámení cesty obsahuje identifikátor zdrojového směrovače (déle označované jako router-id). FD udržována směrovači se pak nevztahuje k samotnému prefixu sítě, ale ke zdroji informace, který je definován jako dvojice (router-id, prefix). Díky tomu Babel garantuje bezsmyčkovost topologie pro každý zdroj, ne pro prefix. Jelikož výsledek sjednocení více acyklických grafů není obecně acyklický, nemůže být při propagaci sítě více zdroji garantována bezsmyčkovost. Možné smyčky jsou však odstraněny nejpozději během doby úměrné velikosti smyčky.

Pro demonstraci uvažujme topologii na obrázku Obrázek 2.8, ve které uzel A zvolil výchozí cestu přes S, a uzel B cestu přes S‘:

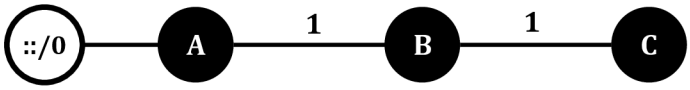


Obrázek 2.8: Topologie s vícenásobnou propagací výchozí cesty

Předpokládejme, že obě výchozí cesty selžou ve stejný okamžik. Pak nic nezabrání tomu, aby A použilo cestu přes B, a současně B použilo cestu přes A. Jakmile A oznámí svou nově používanou cestu uzlu B, přestane cesta přes A pro uzel B splňovat FC. Analogicky, až B oznámí novou cestu uzlu A. Je patrné, že směrovací smyčka zmizí, jakmile aktualizační oznámení projde celou smyčkou. Jelikož může být tento proces ovlivněn ztrátou paketů při přenosu, snaží se Babel v tomto případě zajistit spolehlivé doručení aktualizací. Navíc, poté co směrovače oznamovaly obě cesty, jsou jim známy oba zdroje. To umožňuje zabránit znovuvytvoření dané smyčky.

2.3.6 Překrývající se prefixy

Doposud byl brán v úvahu případ předpokládající disjunktnost všech prefixů v topologii. V praxi ovšem může docházet k překrývání. Speciálním příkladem je výchozí cesta, překrývající se se všemi ostatními prefixy. Po selhání cesty, není obecně správné použít jako její náhradu cestu k síti s kratším prefixem, zahrnujícím prefix původní. Tuto skutečnost demonstrujme na topologii zobrazené obrázkem Obrázek 2.9:



Obrázek 2.9: Topologie pro demonstraci vytvoření smyčky díky překryvu prefixů

Uvažujme selhání uzlu C. Jestliže by B přeposlalo paket určený C pomocí výchozí cesty, vznikla by směrovací smyčka, přetrvávající do doby, než se A od B dozví o odvolání přímé cesty do C. Babel tomuto stavu předchází udržováním „*nedostupné*“ cesty v paměti určitou dobu po jejím odvolání. Tato doba by měla odpovídat nejhoršímu možnému případu propagace odvolání cesty.

2.4 Činnost protokolu

Tato kapitola popisuje činnost protokolu Babel. Není-li uvedeno jinak, pojmem adresa používaným dále v textu, se rozumí adresa na síťové vrstvě referenčního modelu ISO/OSI [20].

2.4.1 Zasílání a příjem zpráv

Zprávy protokolu Babel využívají pro přenos *UDP* pakety. Každá zpráva se skládá z jedné nebo více datových položek, tzv. *TLV* (Type-Length-Value). Každá takováto položka je tvořena třemi částmi. První určuje typ datové položky, druhá její délku a třetí samotnou hodnotu. Zdrojová adresa síťové vrstvy paketu je vždy jednosměrová adresa (unicast). Přesněji jednosměrová linková-lokální (link-local), v případě protokolu IPv6. Cílovou může být buď určená skupinová, nebo jednosměrová adresa. Společnost IANA přidělila protokolu Babel pro IPv6 skupinu *ff02::1:6*, a pro IPv4 skupinu *224.0.0.111*. Jako číslo UDP protokolu bylo zaregistrováno *6696*. To je ve zprávách použito pro zdrojový i cílový port. S vyjímkou TLV Hello a Ack, mohou být všechny TLV zasílány jako unicast i multicast, a jejich význam nijak nezávisí na způsobu zaslání. Zprávy jsou odesílány s mírným zpožděním, během kterého jsou datové položky uchovávány v dočasném úložišti, a to ze dvou důvodů. Zaprvé, díky tomu se lze vyhnout synchronizaci uzlů napříč sítí, a zadruhé je tímto umožněno seskupování více TLV do jedné zprávy. Přesná délka zpoždění aplikovaná na zprávy závisí na tom, zda zpráva obsahuje naléhavé TLV.

Existují zvláštní TLV, na jejichž odesílání jsou kladeny speciální nároky. Např. Ack TLV musí být odesláno před vypršením lhůty specifikované v odpovídajícím požadavku.

2.4.2 Datové struktury

2.4.2.1 Identifikátor směrovače

Každý směrovač je v rámci směrovací domény jednoznačně identifikován, a to pomocí identifikátoru *router-id*. Ten je reprezentován jako řetězec 8 bajtů. Je doporučováno přidělovat identifikátor ve formátu *modifikovaného EUI-64*, který definuje způsob určení části IPv6 adresy z linkové adresy.

2.4.2.2 Sekvenční číslo

Sekvenční číslo směrovače je reprezentováno pomocí 16bitového čísla. Slouží pro vyjádření čerstvosti informací o cestách. Je vkládáno do všech oznámení o cestách, jejichž zdrojem je daný směrovač. Vždy při přijetí požadavku na zvýšení, je číslo navýšeno (modulo 2^{16}).

2.4.2.3 Tabulka rozhraní

Tabulka obsahující seznam síťových rozhraní, na kterých je aktivní protokol Babel. Každá položka obsahuje *sekvenční číslo Hello* daného rozhraní. To je reprezentováno jako 16bitové číslo a je zasíláno v Hello TLV odeslaných příslušným rozhraním. Je zvyšováno (modulo 2^{16}) při odeslání každého Hello TLV. Toto číslo není nijak závislé na sekvenčním čísle směrovače. S každou položkou tabulky jsou svázány dva časovače. První, *Hello časovač* určující zasílání Hello a IHU TLV, a druhý, *Update časovač*, řídící pravidelné posílání aktualizací cest.

2.4.2.4 Tabulka sousedů

Je tvořena seznamem všech síťových rozhraní sousedních směrovačů, ze kterých byla nedávno přijata zpráva protokolu Babel. Položky obsahují informace:

- **Lokální síťové rozhraní**, přes které je daný soused dostupný,
- **adresu rozhraní souseda**,
- **historii přijatých Hello TLV** od daného souseda,
- **cenu odesílání TXCOST**, což je hodnota z poslední přijaté IHU TLV (v případě nepřijetí v dané lhůtě obsahuje „nekonečno“, reprezentované jako *0xFFFF*) vyjadřující hodnocení kvality linky a
- **Hello sekvenční číslo** souseda, očekávané v následujícím Hello TLV.

Položky jsou indexovány dvojicí (lokální rozhraní, adresa souseda). Proto je sousedství vztahem mezi rozhraními, nikoli mezi směrovači. Díky tomu mohou dva směrovače s více rozhraními mezi sebou vytvořit více vztahů sousedství. Ke každé položce jsou přiřazeny dva časovače. Hello časovač,

inicializovaný na interval z přijaté Hello TLV, sloužící pro detekci ztráty Hello zprávy. Dále pak IHU časovač, inicializovaný na malý násobek intervalu neseným v IHU TLV.

2.4.2.5 Tabulka zdrojů

Slouží pro uchovávání vhodných vzdáleností FD. Položky obsahují:

- **Prefix**, definovaný jako dvojice (prefix, plen), kde prefix je prefix sítě a plen je délka prefixu,
- **identifikátor směrovače**, který je zdrojem informace o tomto prefixu,
- **FD**, definovaná jako dvojice *(s, m)*, kde *s* značí sekvenční číslo a *m* metriku.

Tabulka je indexována trojicí (prefix, plen, router-id). Ke každé položce je vázán *časovač úklidu GC*, který bývá inicializován na dobu v řádu jednotek minut.

2.4.2.6 Tabulka topologie

V dokumentu [1] je tato struktura pojmenována jako *tabulka cest* (Route table), který je snadno zaměnitelný s pojmem *směrovací tabulka* (Routing table). Z tohoto důvodu je v této práci pro její označení používán pojem *tabulka topologie* (Topology table), zkracovaný jako TT, používaný např. ve směrovacím protokolu EIGRP.

Obsahuje cesty známé danému směrovači. Každá položka je tvořena následujícími daty:

- **Zdroj informace**, definovaný jako trojice (prefix, plen, router-id), kterým je tato informace propagována,
- **adresa sousedního směrovače** oznamujícího tuto cestu,
- **metrika**, se kterou byla cesta oznámena sousedním směrovačem (tzv. Reported Distance, RD), nebo 0xFFFF pro nedávno odvolanou cestu,
- **sekvenční číslo**, se kterým byla cesta oznámena,
- **adresa následujícího směrovače** na cestě k cíli,
- **binární příznak**, indikující zda je cesta zvolena tj. aktuálně používána pro přeposílání a oznamována sousedním směrovačům.

Tabulka je indexována trojicí (prefix, plen, neighbour). S položkami je asociován *Route časovač,* kontrolující platnost dané informace.

2.4.2.7 Tabulka nevyřízených požadavků

Tabulka obsahující seznam požadavků na zvýšení sekvenčního čísla odeslaných daným směrovačem, na které nebyla dosud přijata odpověď. Seznam zahrnuje jak požadavky pocházející od tohoto směrovače, i požadavky přeposlané. Položky jsou tvořeny z:

- **Prefixu sítě**,
- **identifikátoru směrovače**,

- žádané hodnoty **sekvenčního čísla**,
- **adresy souseda**, od nějž je původně doručen přeposílaný požadavek,
- **počtu skoků**, udávající kolikrát může být ještě požadavek přeposlán.

Ke každému nevyřízenému požadavku se vztahuje jeden časovač, sloužící jak pro znovu odeslání, tak i vypršení platnosti.

2.4.3 Potvrzované zprávy

Směrovač odesílající zprávu může požadovat, aby během určité doby přijímající směrovač explicitně potvrdil doručení zprávy. Přestože požadování potvrzení je volitelné, každý směrovač musí být schopen na takovýto požadavek odpovědět. Požadavek je možné odeslat jako unicast, tak i jako multicast, kdy je odpověď požadována od všech příjemců. Odpověď musí být odeslána vždy jako unicast.

2.4.4 Získávání sousedů

Získávání sousedů je proces, během kterého je směrovačem objevována množina sousedních směrovačů a všech jejich síťových rozhraní zahrnutých ve směrování protokolem Babel. Při procesu je též zjišťována obousměrná dostupnost. Navíc poskytuje statistiky, které mohou být použity při výpočtu kvality linky. Směrovače úspěšně naváží vztah sousedství i v případě, že používají rozdílné parametry časovačů.

2.4.5 Detekce dostupnosti

Směrovač pravidelně odesílá na všech svých síťových rozhraních zahrnutých v procesu směrování Hello zprávy. Hello TLV nesou sekvenční číslo Hello a interval odesílání těchto zpráv. Navíc může směrovač odeslat i neplánovanou Hello zprávu, a to je-li objeven nový soused, či došlo-li k náhlé změně parametrů linky. Při příjmu nebo zjištění ztráty zprávy, je uzlem přepočítána cena linky a spuštěna procedura výběru cesty.

Pro detekci kvality linky v opačném směru, směrovač pravidelně zasílá zprávy obsahující datové položky IHU (“I Heard You“) TLV. Protože IHU TLV obsahuje vlastní interval zasílání, nezávislý na intervalu Hello, mohou být zasílány méně často než zprávy Hello, což je i doporučováno. Přestože je IHU TLV koncepčně jednosměrové, mělo by být posíláno jako multicast, díky čemuž nevyvolá proces překladu mezi adresami síťové a linkové vrstvy (ARP a ND). Dále IHU TLV nese informaci ceny příjmu RXCOST z pohledu odesílatele zprávy, která je příjemcem zahrnuta do výpočtu ceny linky. Při příjmu této zprávy je příjemcem v odpovídající položce tabulky sousedů aktualizována položka ceny odesílání TXCOST a resetován IHU časovač, na malý násobek IHU intervalu. Vyprší-li lhůta pro

doručení IHU zprávy, je TXCOST daného souseda nastaveno na nekonečno. Po aktualizaci TXCOST, je uzlem přepočítána cena linky a spuštěna procedura výběru cesty.

2.4.6 Výpočet ceny linky

Cena přenosu linkou mezi sousedními směrovači je vypočítávána na základě hodnot udržovaných v tabulce sousedů. Jedná se o hodnotu RXCOST, odvozené z historie příjmu Hello zpráv, a hodnotu TXCOST, získávanou od souseda ve zprávách IHU. TXCOST je ve skutečnosti sousedova hodnota RXCOST. Jak přesně tyto hodnoty ve výpočtu ceny linky použít, protokol Babel nespecifikuje, čímž umožňuje použití rozdílných způsobů. Ovšem pro zajištění správné funkčnosti protokolu, je nutné dodržet následující podmínky:

- Cena je vždy pozitivní,
- jestliže v nedávné době nebyly přijaty zprávy Hello, je cena nekonečná,
- jestliže je TXCOST nekonečná, pak je i cena nekonečná.

2.4.6.1 K-out-of-j

K-out-of-j je způsob výpočtu ceny linky, vhodný pro použití na drátových linkách. Tyto linky jsou buď funkční, což je stav, ve kterém dochází ke ztrátě paketu pouze zřídka, nebo nefunkční, kdy jsou linkou zahazovány všechny pakety.

Strategie *k-out-of-j* je parametrizována konstantou jmenovité ceny linky, označovanou C, pro niž platí $C \geq 1$, a dvěmi malými čísly k a j, pro které platí $0 < k \leq j$. Směrovač udržuje historii příjmu posledních j zpráv. Jestliže k a více zpráv bylo úspěšně přijato, je linka považována za funkční a RXCOST je nastaveno na C. V opačném případě je linka považována za nefunkční, a RXCOST je nastaveno na nekonečno. Cena linky je definována následovně:

$$cena = \begin{cases} \infty & pokud\ RXCOST = \infty \\ TXCOST & jinak \end{cases} \tag{2.3}$$

2.4.6.2 ETX

Způsob výpočtu ceny linky ETX (Estimated Transmission Cost, [25]) je založen na odhadu počtu znovu odeslání zprávy, nutného pro úspěšné doručení. Tento způsob je vhodný pro použití v bezdrátových sítích. Z historie příjmu Hello zpráv je vypočítán odhad pravděpodobnosti úspěšného příjmu zpráv Hello, označovaný jako β. *RXCOST* je definován jako

$$RXCOST = \frac{256}{\beta} \tag{2.4}$$

Na základě znalosti definice *RXCOST*, definujme i hodnotu odhadu pravděpodobnosti úspěšného odeslání zprávy Hello, označeného α, a to následovně

$$\alpha = \min\left(1, \frac{256}{TXCOST}\right) \tag{2.5}$$

Výpočet je pak určen vztahem

$$cena = \frac{256}{\alpha * \beta} \tag{2.6}$$

kdy po dosazení a úpravě dostáváme

$$cena = \frac{max(TXCOST, 256) * RXCOST}{256} \tag{2.7}$$

2.4.7 Podmínka vhodnosti

Podmínka vhodnosti FC je použita na všechny přijaté Update TLV, s cílem filtrovat oznámení, jež by mohly způsobit vytvoření směrovací smyčky. Jak již bylo popsáno v kapitole 2.3.3, FC porovnává vzdálenost nesenou přijatým oznámením RD s hodnotou FD daného směrovače. Oznámení s konečnou vzdáleností tak velkou, že ohrožují bezsmyčkovost topologie, jsou ignorovány.

Vhodná vzdálenost FD je dvojice (sekvenční číslo, metrika), kde sekvenční číslo je 16bitové číslo (modulo 2^{16}) a metrika je nezáporné 16bitové číslo. Říkáme, že vzdálenost *(s, m)* je striktně lepší než vzdálenost *(s', m')*, zapisováno jako $(s, m) < (s', m')$, pokud platí

$$s > s' OR\ (s = s' AND\ m < m') \tag{2.8}$$

kdy sekvenční čísla jsou porovnávána modulo 2^{16}.

Přijatá aktualizace vyhovuje podmínce FC, je-li oznamovaná vzdálenost striktně lepší než FD, nebo v případě kdy jde o odvolání cesty (metrika nastavena na nekonečno). Přesněji musí splňovat následující:

- Metrika je nekonečno, nebo
- v tabulce zdrojů neexistuje položka pro daný zdroj, nebo
- v tabulce zdrojů existuje položka pro daný zdroj, ovšem oznámení nese vzdálenost striktně lepší než FD, jak je definováno výše.

2.4.8 Výpočet metriky

Metrika je počítána z metriky oznamované sousedním směrovačem a ceny linky k danému sousedu. Stejně jako u ceny linky, umožňuje Babel použití různých způsobů výpočtu. Funkci pro výpočet metriky označme *M(c, m)*, kde *c* značí cenu linky a *m* metriku cesty oznámenou sousedem. Takováto funkce musí splňovat následující podmínky:

- Je-li *c* nekonečno, je i výsledek *M(c, m)* nekonečno,
- M je *ryze monotónní*: $M(c,m) > m$.

Navíc by metrika měla splňovat vlastnost:

- M je *isotónní*: $Jestliže\ m \leq m', pak\ M(c,m) \leq M(c,m')$.

Ryzí monotónnost funkce je pro zajištění topologie bez smyček nezbytná. *Isotónnost* nezbytná není, ovšem bez jejího splnění nemusí být výsledná topologie globálním optimem.

Metriku splňující obě tyto vlastnosti lze jednoduše definovat například jako sumu cen všech linek zahrnutých v dané cestě, zapsanou $M(c,m) = c + m$.

2.5 Existující implementace

2.5.1 Babeld

Ukázková implementace vyvíjená od roku 2007 samotným autorem protokolu J. Chroboczekem. Dostupná jako open source pod licencí MIT[1]. Běžící jako samostatný démon, kompatibilní s operačními systémy založenými na GNU/Linux a BSD. Volně ke stažení v podobě zdrojových kódů [19], či předkompilovaných balíčků. Taktéž je tato implementace dostupná i jako modul ve vývojové verzi softwarového směrovače Quagga [18]. Pro uzly připojující koncové sítě, tzv. stub směrovače, existuje minimalistická verze, dostupná pod názvem *sbabeld*.

Díky jeho flexibilitě a oblíbenosti v řadách výzkumníků, je *babeld* stále vyvíjen a rozšiřován o podporu nových mechanismů umožňujících jeho nasazení v rozmanitých prostředích. Verze 1.5.1, aktuální v době psaní textu, implementuje protokol Babel verze 2. Umožňuje redistribuci a pokročilou filtraci cest. Pro různé typy sítí nabízí způsoby výpočtu metrik zohledňující rozdílné faktory, a to počet skoků, odhad ceny přenosu, rádiové rušení a dobu přenosu linkou. Zahrnuje algoritmus výběru cest zohledňující historii metrik cest, čímž preferuje stabilní cesty. Dosud nepodporuje přenos po IPv4 a zabezpečení autentizací.

[1] Celé znění dostupné na http://opensource.org/licenses/MIT

Program je spouštěn příkazem `babeld VOLBY [--] SITOVA-ROZHRANI`, např. `babeld -D wlan0 wlan1`. Konfigurace je možná buď pomocí parametrů programu, nebo konfiguračními soubory. Výchozím konfiguračním souborem je */etc/babeld.conf*. Klíčová slova používaná v konfiguračních souborech jsou, včetně jim odpovídajícím parametrům programu, uvedena v příloze v tabulce Tabulka 8.

2.5.2 TinyOS rozšíření

Zjednodušená implementace, která vznikla v rámci projektu HydroNet, zabývající se návrhem a vývojem nových technických platforem pro monitorování kvality vody. Vyvíjená platforma využívá plovoucí bóje a bezpilotní vory vybavené senzory měřící znečištění. Pro řízení a komunikaci používají rádiové spoje na frekvenci 434 MHz. Jelikož prvky sítě nejsou pouze zdroji informací, potřebují vzájemně komunikovat. To díky vzdálenostem a rušení není vždy možné uskutečnit přímo mezi libovolnými dvěmi prvky. Zařízení jsou nízko-energetická s velmi omezenými prostředky, využívající operačním systémem *TinyOS*. Z tohoto důvodu jsou na efektivitu směrovacího protokolu kladeny velké nároky.

Implementace pro operační systém *TinyOS* je oproti návrhu protokolu v dokumentu [1] velmi zjednodušena a přizpůsobena přímo pro dané použití. Například, jelikož prvky disponují pouze jedním síťovým rozhraním, stává se tabulka rozhraní zbytečnou. Dále je, díky zjednodušené hierarchii adresace v síti, vynechána podpora směrování na základě prefixu. A následuje mnoho dalších zjednodušení a úprav pro zefektivnění procesu směrování.

Na příkladu této implementace je patrné, že základní koncept protokolu je vhodné použít i ve velmi specifických oblastech.

3 Prostředí OMNeT++

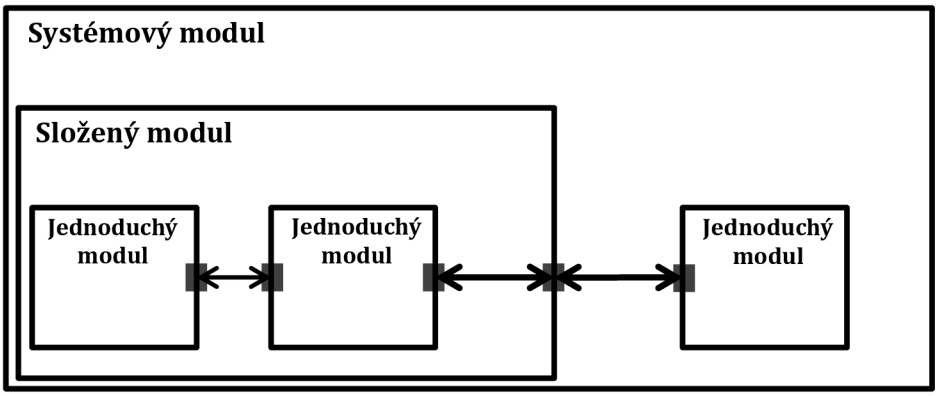
OMNeT++ je objektově orientovaný, modulární, diskrétní simulátor, který poskytuje prostředí a nástroje pro vytváření simulací. Díky obecné architektuře, může být použit v různých oblastech, a to nejčastěji pro modelování drátových a bezdrátových počítačových sítí, modelování komunikačních protokolů, ovšem také pro modelování víceprocesorových systémů, validaci hardwarových architektur, vyhodnocování výkonnosti softwarových systémů a dalších. Obecně lze říci, že je možné simulovat libovolný systém, který lze považovat za diskrétní a lze jej reprezentovat objekty komunikujícími výměnnou zpráv. Moduly specifické pro určité obory, jsou poskytovány v rámci frameworků, vyvíjené jako samostatné projekty. OMNeT++ podporuje běh simulací jak v grafickém uživatelském prostředí, užitečném pro demonstrační a ladící účely, tak i v prostředí příkazového řádku, vhodném pro dávkové zpracovávání. Simulátor je snadno přenositelný mezi nejběžněji používanými operačními systémy (Microsoft Windows™, GNU/Linux, Apple Mac OS X). Také je možné využít paralelní distribuovanou simulaci, a to díky podpoře mechanismů pro komunikaci distribuovaných systémů (např. MPI). Kromě simulačního jádra nabízí OMNeT++ také vývojové prostředí, využívající Eclipse IDE. OMNeT++ je volně dostupný pro akademické, vzdělávací a neziskové účely. Pro komerční použití je nutné využít verzi zvanou OMNEST.

Diskrétní systém je systém, ve kterém změny probíhají skokově a netrvají žádnou dobu. Simulace diskrétního systému reprezentuje změny pomocí událostí. Událost je určena časem jejího výskytu a prioritou. Čas, kdy dochází k události, se nazývá aktivační čas. Priorita určuje, která událost nastane dříve v případě, že je naplánováno více událostí ve stejný čas. Časová osa simulovaného systému se nazývá simulační čas, nebo také modelový čas. Mezi dvěmi událostmi se simulovaný systém nemění. Pro implementaci diskrétní simulace využívá simulátor datovou strukturu *kalendář událostí* (Future Event List).

Při spuštění simulace je nejprve inicializován simulační model a do kalendáře jsou vloženy události. V průběhu simulace jsou události z kalendáře vybírány. Pro zajištění kauzality je vždy vybírána událost s nejnižším aktivačním časem. Je-li na stejný čas naplánováno více událostí, je vybrána událost s vyšší prioritou. Pokud se shodují i priority událostí, je vybrána dříve naplánovaná událost. Po vybrání události je aktuální simulační čas změněn na aktivační čas události a událost provedena. V OMNeT++ se provedením události rozumí zaslání zprávy mezi moduly simulačního modelu – událost je zpracována v cílovém modulu uživatelsky definovaným kódem. Při provádění události můžou být do kalendáře plánovány nové události, nebo již naplánované mazány. Simulace končí, jestliže je kalendář událostí prázdný nebo byl dosažen limit simulačního času.

3.1 Struktura modelu

Simulační model je sestaven z hierarchicky vnořených modulů, které mezi sebou komunikují pomocí zasílání zpráv. Modul nejvyšší úrovně se v OMNeT++ nazývá *systémový modul* (system module), a obsahuje podmoduly, které taktéž mohou obsahovat podmoduly. Hloubka vnořování modulů je neomezená, což umožňuje věrně popsat logickou strukturu libovolného modelovaného systému. Struktura modelu je popsána pomocí jazyka NED (Network Description). Modul obsahující podmoduly se nazývá *složený modul* (compound module). Opakem je *jednoduchý modul* (simple module) vyskytující se na nejnižší úrovni hierarchie a neobsahující žádné vnořené moduly. Jednoduché moduly implementují chování modelu, které je popsáno jazykem C++. Při popisu modelu uživatel definuje *modulové typy* (module types) – instance těchto typů vystupují jako části složitějších modulových typů. Oba typy modulů, jednoduché i složené, jsou instancemi modulových typů. Na nejvyšší úrovni stojí systémový modul jako instance již definovaných modulových typů. Koncept jednoduchých a složených modulů je podobný přístupu u formalismu DEVS, definující *atomické* (atomic models) a *spojované modely* (coupled models). Koncept modulů umožňuje jednoduché znovupoužití jednotlivých částí při definování nových modulů.



Obrázek 3.1: Hierarchie modulů v prostředí OMNeT++

Komunikace mezi moduly probíhá přes komunikační rozhraní zvané *brána* (gate). Existují tři typy bran: *vstupní*, přes které jsou zprávy modulem přijímány, *výstupní*, přes které jsou zprávy modulem odesílány a *vstupně-výstupní*, sloužící pro odesílání i přijímání. Zasílané zprávy mohou obsahovat složité datové struktury. Zprávy jsou po odeslání rozhraním přeneseny spojeními mezi moduly a doručeny cílovému modulu. Modul může poslat zprávu i sám sobě, tzv. *self-messages*, často využívané pro implementaci časovačů. Každé spojení, taktéž nazývané linka, je vytvářeno v rámci jedné úrovně hierarchie modelu. Ve složeném modulu je možné propojit odpovídající brány dvou podmodulů, nebo bránu jednoho podmodulu s bránou složeného modulu. Propojení napříč různými úrovněmi hierarchie není dovoleno, jelikož by znemožnilo znovupoužití modulů. Díky hierarchické struktuře modelu, zprávy typicky putují skrze posloupnost linek, začínající a končící jednoduchými

moduly. Složené moduly zprávy pouze transparentně předávají mezi vnějšími a vnitřními linkami modulu. Jednotlivým linkám mohou být nastaveny parametry přenosu jako zpoždění, rychlost a chybovost.

3.2 INET

INET Framework je open-source balík modelů prvků počítačových sítí určený pro simulační prostředí OMNeT++. Obsahuje modely drátových i bezdrátových protokolů linkových vrstev (Ethernet, PPP, 802.11), síťových a transportních protokolů (IP, IPv6, TCP, UDP, SCTP), směrovacích protokolů (RIP, OSPF) a mnoha dalších mechanismů. Pro popis modelu je využit koncept modulů, komunikujících zasíláním zpráv. Koncové stanice, směrovače, přepínače a další počítačové prvky jsou reprezentovány složenými moduly. Tyto složené moduly jsou tvořeny z jednoduchých modulů implementujících funkcionalitu jednotlivých protokolů a aplikací. Moduly jsou organizovány v hierarchii balíčků, která přibližně odpovídá referenčnímu modelu ISO/OSI [20], kde na nejvyšší úrovni se nacházejí balíčky inet.linklayer, inet.networklayer, inet.transport a inet.applications. Dále také inet.base, inet.util, inet.world, inet.mobility a inet.nodes. Dané hierarchii odpovídá i stromová struktura zdrojových kódů modulů, např. pro balík inet.applications se jedná o adresář src/applications. Moduly jsou definovány v souborech *.NED*. Tyto soubory konfigurují parametry modulu, jeho podmoduly a spojení s ostatními moduly. Popis funkcionality jednoduchého modulu se nachází ve třídě jazyka C++, pojmenované shodně s modulem, jež implementuje. Formát zpráv a hlavičky komunikačních protokolů jsou definovány v souborech *.MSG*, z jejichž obsahu jsou nástrojem *opp_msgc* generovány C++ třídy. Balíček inet.nodes obsahuje předpřipravené modely síťových prvků použitelné v simulaci, např. model koncové stanice StandardHost, model směrovače Router, model přepínače EtherSwitch, model bezdrátového přístupového bodu WirelessAP a další. Děděním těchto modelů lze velmi usnadnit tvorbu nových specifických modelů.

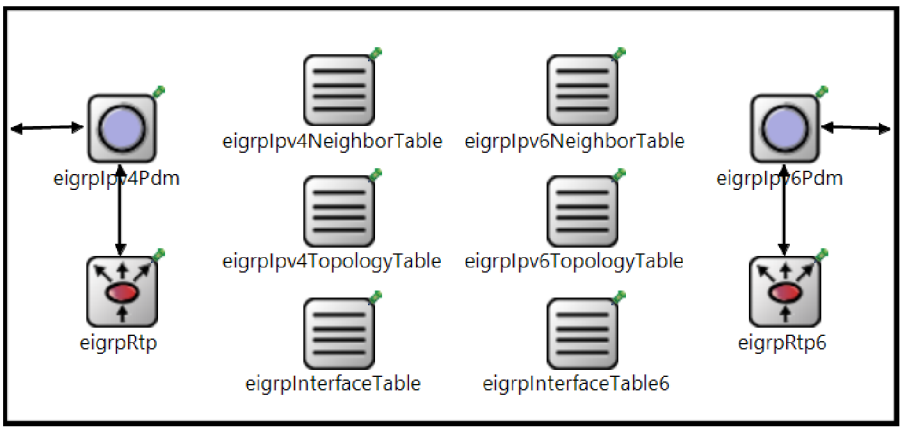
3.2.1 ANSAINET

Na Fakultě informačních technologií Vysokého učení technického v Brně řeší výzkumná skupina NES@FIT projekt ANSA (Automated Network Simulation and Analysis), zabývající se automatizovanými metodami vytváření simulačních modelů, formální analýzy a verifikace počítačových sítí na základě znalosti topologie a konfigurace síťových zařízení. V rámci tohoto projektu vzniklo rozšíření frameworku INET, zvané *ANSAINET*. Toto rozšíření přidává podporu protokolů PIM-SM, VRRPv2, OSPFv3, EIGRP, RIPng, TRILL, IS-IS a dalších užitečných modulů, např. DeviceConfigurator pro načítání konfigurací zařízení ze souboru XML, či AnsaIPTrafGen umožňující pokročilé generování síťového provozu.

3.3 EIGRP pro IPv6

Jelikož jsem se simulováním v OMNeT++ a s vývojem pro ANSAINET neměl dosud žádné zkušenosti, bylo mi mým vedoucím práce navrhnuto podílet se nejprve na vývoji již existujícího modelu. Mým úkolem bylo rozšířit model směrovacího protokolu EIGRP o podporu protokolu IPv6.

Existující implementace podporovala pouze protokol IPv4. Aby zároveň podporovala oba protokoly, byly zduplikovány podmoduly modulu EigrpProces, kde duplikáty slouží pro IPv6. V EIGRP je podpora různých protokolů síťové vrstvy rozdělena do tzv. protokolově závislých modulů (PDM, Protocol Dependent Modules). Proto byl přidán nový modul EigrpIpv6Pdm. Výslednou strukturu znázorňuje Obrázek 3.2. Podpůrné třídy modulu EigrpIpv4Pdm nebyly připravené pro univerzální použití s jiným síťovým protokolem. Abych se vyvaroval duplikaci těchto tříd na úrovni zdrojového kódu, využil jsem techniku metaprogramování, jež je jazykem C++ podporována. Existující třídy byly zobecněny na šablony. Použití šablon ovšem přináší v prostředí OMNeT++/INET určitá úskalí. Jak již bylo zmíněno v kapitole věnující se frameworku INET, třída implementující funkcionalitu jednoduchého modulu se musí s tímto modulem jmenovat shodně. To však se složitým zápisem instance šablony, využívající speciální znaky, není možné dodržet. Řešením tohoto problému je využití jiné vlastnosti jazyka C++, a sice dědičnosti. Třída s patřičným názvem je pak přímým potomkem instance šablony.



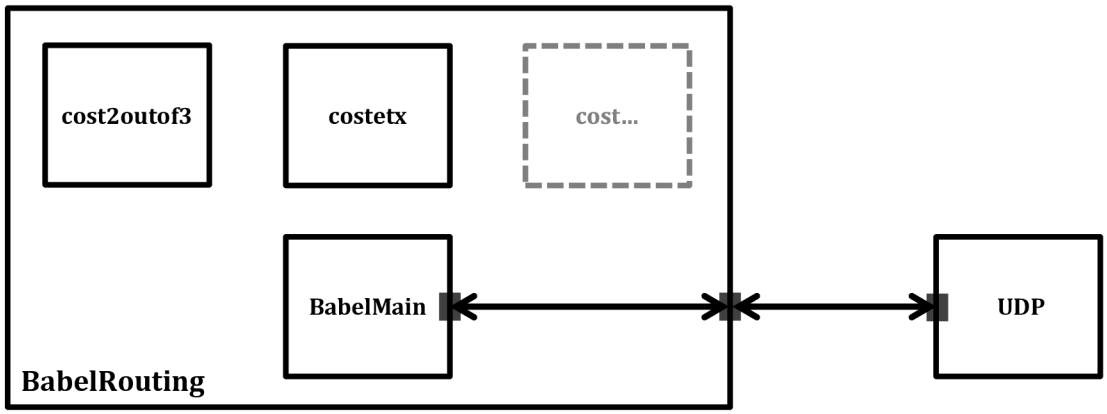
Obrázek 3.2: Výsledný modul EIGRP procesu, podporující IPv4 i IPv6

Nejzásadnějším rozdílem z pohledu implementace směrovacího protokolu mezi IPv4 a IPv6 je počet logických sítí, do kterých může síťové rozhraní náležet. V IPv4 je to striktně jedna logická síť, v IPv6 však počet není omezen. S tímto souvisí i rozdíl způsobu konfigurace, kdy v IPv6 je povolován běh protokolu pouze přímo na síťovém rozhraní, nikoli uvedením adresy sítě. Další odlišností je použití linkových-lokálních IPv6 adres v EIGRP zprávách. Tyto a další rozdílnosti jsou implementací zohledněny.

4 Návrh implementace

V této kapitole je navržen simulační model směrovacího protokolu Babel. Bude implementován jako složený modul v prostředí OMNeT++/INET, jehož struktura je znázorněna na obrázku Obrázek 4.1.

Je tvořen hlavním modulem *BabelMain* implementující samotné chování protokolu, a moduly pro výpočet ceny linky. V původním návrh implementace byl modul *BabelMain* modelován jako složený modul tvořený několika jednoduchými moduly, reprezentující abstraktní datové typy. Od tohoto záměru bylo upuštěno, jelikož díky složitému propojení jednotlivých struktur není dost dobře možné rozsáhlejší změny provést pouze v jedné struktuře, čímž modularita na úrovni OMNeT++ postrádá smysl. Kde naopak smysl má, je využití pro reprezentaci rozdílných způsobů výpočtu ceny linky. Přidání či změnu parametrů je pak možné provést jednoduše přidáním, respektive změnou, modulu v souboru .NED, bez znalosti jazyka C++ a nutnosti kompilace zdrojových kódů. V základu implementace obsahuje tyto moduly dva. Prvním je *cost2outof3*, používající pro výpočet metodu *k-out-of-j* s parametry $k = 2$ a $j = 3$, standardně využívanou na drátových linkách. Druhým je *costetx* implementující metodu *ETX*, užívanou na bezdrátových spojích.



Obrázek 4.1: Návrh struktury složeného modulu, implementujícího funkcionalitu protokolu Babel

Implementace klade velký důraz na shodné kódování a kompresi přenášených zpráv s dokumentem [1]. Díky tomu jsou výstupem simulace informace o využívání přenosové kapacity, které mohou být zajímavé při porovnání s jinými směrovacími protokoly. Je podporován přenos informací o cestách protokolů IPv4 i IPv6, nezávisle na použitém protokolu, a to jak po IPv6, tak i po IPv4. Tímto se implementace liší od *babeld*, která přenos po IPv4 nepodporuje.

4.1 Abstraktní datové typy

V kapitole 2.4.2 jsou uvedeny datové struktury, nutné pro běh protokolu. Tyto struktury využívá modul *BabelMain.* Pojmenovány jsou následovně:

- *BabelInterfaceTable* – implementující tabulku rozhraní, obsahující seznam síťových rozhraní, na kterých je protokol aktivní.
- *BabelNeighbourTable* – datový typ tabulky sousedů, uchovávající seznam síťových rozhraní sousedních směrovačů, se kterými byl navázán vztah sousednosti. Zahrnuje data používaná při výpočtu kvality linky.
- *BabelTopologyTable* – tabulka topologie, obsahující všechny cesty známé danému směrovači.
- *BabelSourceTable* – tabulka zdrojů informací, obsahující seznam síťových prefixů a identifikátory směrovačů, kterými jsou propagovány. Slouží pro uchování hodnot FD.
- *BabelPenSRTable* – seznam nevyřízených požadavků. Struktura používaná při přeposílání požadavků na navýšení sekvenčního čísla.
- *BabelFtlv* – struktura pro interní reprezentaci nekomprimované položky TLV.
- *BabelBuffer* – struktura pro dočasné uložení datových položek *BabelFtlv*, před jejich odesláním. Díky ní může být do odesílaných zpráv shlukováno více TLV, což vede k zefektivnění přenosu zpráv.

První návrh implementace počítal s použitím stejného způsobu řešení podpory více rodin protokolů jako při rozšiřování modelu EIGRP, a to odlišením do vlastních struktur daného protokolu, jak je popsáno v kapitole 3.3. Ovšem protože architektury obou protokolů jsou značně odlišné, toto řešení pro protokol Babel není vhodné. Jako vhodnější se ukázalo využít univerzální datový typ, umožňující reprezentaci IPv4 i IPv6 adresy, a položky pro různé protokoly ukládat v jedné datové struktuře.

4.2 Konfigurace

Modul umožňuje načítání konfigurace ze souboru ve formátu XML. Návrh se snaží vycházet z v praxi již používaných principů. Proto využívá stejná klíčová slova, definovaná implementací *babeld*, o která rozšiřuje strukturu používanou modulem *DeviceConfigurator*.

Spuštění běhu protokolu

Protokol je na směrovači aktivován následující částí konfigurace:

```
<Routing>
    <Babel>
        <RouterId>...</RouterId>
        <Port>...</Port>
    </Babel>
</Routing>
```

- *RouterId* - Explicitní specifikace identifikátoru směrovače, která je volitelná. Při neuvedení je identifikátor určen automaticky z IPv6 adresy prvního síťového rozhraní ve formátu modifikovaného EUI-64, případně vygenerován náhodně.
- *Port* – číslo UDP portu využívaného pro odesílání a příjem zpráv. Bez uvedení je použit port 6696.

Aktivace protokolu na síťovém rozhraní

Síťové rozhraní je do procesu směrování zahrnuto pomocí:

```
< Interfaces>
    <Interface name="...">
      <Babel>
        < SplitHorizon>...</SplitHorizon>
        <Rxcost>...</Rxcost>
        <HelloInterval>...</HelloInterval>
        <UpdateInterval>...</UpdateInterval>
        <Wired>...</Wired>
        <AFSend>...</AFSend>
        <AFDistribute>...</AFDistribute>
        <CostCompModule>...</CostCompModule>
      </Babel>
    </Interface>
</ Interfaces>
```

kde jednotlivé položky mají následující význam:

- *SplitHorizon* – Určuje použití mechanismu Split-Horizon daného rozhraní. Bez uvedení je na drátových rozhraních aktivní, na bezdrátových neaktivní.
- *Rxcost* – Specifikuje cenu příjmu zpráv za ideálních podmínek. Bez uvedení je na drátových rozhraních nastavena na 96, na bezdrátových 256.
- *HelloInterval* – Interval zasílání pravidelných Hello zpráv, v jednotkách desítek milisekund. Bez uvedení nastaven na 20 s na drátových rozhraních, 4 s na bezdrátových.
- *UpdateInterval* – Interval zasílání pravidelných Update zpráv, v jednotkách desítek milisekund. Bez uvedení je nastaven na čtyřnásobek hodnoty HelloInterval.
- *Wired* – Specifikace typu daného rozhraní. Bez uvedení je typ zjištěn automaticky.
- *AFSend* – Specifikace rodin protokolů používaných pro odesílání a příjem zpráv. Možné hodnoty jsou:
    - *IPv6* – pouze protokol IPv6, výchozí volba,
    - *IPv4* – pouze protokol IPv4,
    - *IPvX* – protokoly IPv6 i IPv4,
    - *Passive* – dané rozhraní zprávy neodesílá a nepřijímá.
- *AFDistribute* – Specifikace rodin protokolů, jejichž prefixy budou zahrnuty do procesu směrování. Možné hodnoty jsou:
    - *IPv6* – pouze protokol IPv6, výchozí volba,
    - *IPv4* – pouze protokol IPv4,
    - *IPvX* – protokoly IPv6 i IPv4,
    - *None* – nejsou zahrnuty žádné prefixy.
- *CostCompModule* – název modulu implementující výpočet ceny linky. Bez uvedení je na drátových rozhraních použit modul *cost2outof3*, na bezdrátových *costetx*.

5 Implementace

V této kapitole je popsána implementace modelu směrovacího protokolu Babel v prostředí OMNeT++. Ta je realizována v jazyce C++ a rozšiřuje knihovnu ANSAINET. Vychází z návrhu, uvedeného v kapitole 4 a snaží se věrně implementovat funkcionalitu protokolu Babel.

Velký důraz je kladen na přesné kódování a kompresi přenášených zpráv. Pro dosažení tohoto cíle byl navržena interní reprezentace TLV, podrobně popsána níže. Také je dbáno na univerzálnost datových struktur pro použití s rozdílnými protokoly síťové vrstvy. Té je dosaženo využitím univerzálního objektu *IPvXAddress*. Tento přístup se jeví jako vhodný, a vede ke zjednodušení architektury programu a minimalizaci duplicitních částí zdrojového kódu.

Realizace odpočtu intervalů

Protokol Babel pro svou funkcionalitu předepisuje celou řadu různých intervalů. Ty jsou implementovány pomocí časovačů, jež v předem naplánovaný časový okamžik (odpovídající konci intervalu) vyvolají událost, ohlašující vypršení stanovené doby. V prostředí OMNeT++ jež je specifické komunikací výměnou zpráv, jsou časovače realizovány jako zprávy odesílané sobě samému. Pro snadnou práci s těmito časovači jsou vytvořeny funkce pro jejich vytvoření, přečasování a smazání. Tyto zprávy mohou mít přiděleny kategorii a kontext. Obě tyto vlastnosti jsou v implementaci využity. Dle kategorie je určen typ kontextu. Kontext je ve skutečnosti ukazatel na objekt, ke kterému daný časovač náleží. Příkladem budiž časovač kategorie HELLO s kontextem objektu síťového rozhraní, na kterém při jeho vypršení má být odeslána zpráva Hello. Pro zjednodušení bývá kontext časovači nastavován již při konstrukci souvisejícího objektu. Celkově je v práci používáno jedenáct různých kategorií:

- *HELLO* – odeslání Hello zprávy na daném rozhraní,
- *UPDATE* – odeslání periodických aktualizací na daném rozhraní,
- *BUFFER* – vyprázdnění dočasné paměti,
- *BUFFERGC* – odstranění nepoužívaných dočasných pamětí,
- *TOACKRESEND* – opětovné odeslání zprávy s požadavkem na spolehlivý přenos,
- *NEIGHHELLO* – interval pro doručení Hello zprávy od sousedního směrovače,
- *NEIGHIHU* – interval pro doručení IHU zprávy od sousedního směrovače,
- *ROUTEEXPIRY* – vypršení platnosti záznamu cesty,
- *ROUTEBEFEXPIRY* – blížící se vypršení platnosti záznamu cesty,
- *SOURCEGC* – vypršení platnosti záznamu zdroje,
- *SRRESEND* – opětovné odeslání zprávy s požadavkem na zvýšení sekvenčního čísla.

5.1 Koncept FTLV

Zvlášť velký důraz je protokolem Babel kladen na úsporné využívání přenosového pásma. Z tohoto důvodu jsou TLV před samotným odesláním po určitou dobu uloženy v dočasné paměti. Díky tomu se do jednoho UDP paketu shlukuje více TLV, čímž se značně snižuje počet přenášených UDP paketů. Tento přístup rovněž umožňuje odstranění duplicitních dat napříč všemi TLV v paketu, zajišťující další úsporu přenosového pásma.

Jelikož model protokolu Babel realizovaný v této práci se snaží přenášené zprávy modelovat co nejvěrněji, bylo potřeba navrhnout vhodný způsob interní reprezentace položek TLV. Tím je právě koncept FTLV, ve kterém jsou jednotlivá TLV reprezentována v nekomprimované podobě, se všemi potřebnými informacemi, a navíc jsou rozšířeny o některá další interní data.

Pro všechny typy TLV popisované v dokumentu [1] existují vlastní typy FTLV. Ty jsou potomky abstraktní třídy *BabelFtlv*, která zahrnuje typ a počet odeslání. Dále poskytuje metody pro přístup k těmto položkám a předepisuje metody pro určení délky a převod do komprimované podoby.

5.1.1 Dočasná paměť

Paměť pro dočasné uložení FTLV položek je implementována pomocí třídy *BabelBuffer*. Tato třída zahrnuje cílovou adresu, odchozí rozhraní, seznam FTLV a časovač plánovaného vyprázdnění. Poskytuje metody pro přístup k položkám a pro práci se seznamem FTLV, mezi kterými jsou mimo jiné dotaz, zda je obsažena FTLV určitého typu, či metoda pro nalezení Update FTLV z pohledu komprese nejvhodnější.

Dočasné paměti jsou odlišovány podle cílové adresy a odchozího rozhraní. Jsou vytvářeny při přijetí požadavku na odeslání objektu FTLV daným rozhraním a cílem v případě, že dosud neexistují. Při vkládání FTLV do paměti je ověřováno splnění některých podmínek bránících kolizním situacím. Jestliže podmínky nejsou splněny, je obsah dočasné paměti ještě před vložením FTLV odeslán. Dále je kontrolováno, zda plánované vyprázdnění paměti splňuje požadavky na maximální dobu uložení vkládaného záznamu. Nejsou-li splněny, je vyprázdnění přeplánováno. Standardně je pro FTLV bez zvláštních požadavků určena doba dočasného uložení náhodně, dle vztahu 5.1.

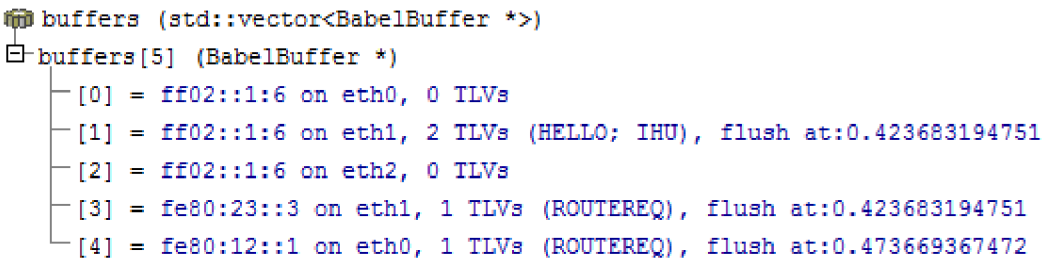
$$BuffFlushInterval \in \left\langle 0{,}75 * \frac{HelloInterval}{4};\ 1{,}25 * \frac{HelloInterval}{4} \right) \tag{5.1}$$

Pro FTLV s požadavkem na urgentní odeslání, je doba stanovena jako

$$BuffFlushInterval \in \langle 0{,}75 * 0{,}2;\ 1{,}25 * 0{,}2) \ s \tag{5.2}$$

Po uplynutí doby uložení, je obsah paměti odeslán, při čemž je provedena komprese obsažených FTLV. Ty jsou nakopírovány do nově vzniklého objektu *BabelMessage*, reprezentující zprávu protokolu Babel. Jelikož pro zajištění spolehlivosti přenosu zpráv existují dvě rozdílné strategie, je během vyprazdňování určeno kolikrát, případně jakým způsobem bude zpráva odeslána. To je provedeno jako stanovení maxima z vyžadovaného počtu odeslání všech obsažených FTLV. V případě, že je ve zprávě zahrnuto FTLV vyžadující odeslání spolehlivým způsobem s použitím požadavku na potvrzení, má tento způsob přednost před vícenásobným odesláním.

Jelikož bude pravděpodobně se stejným cílem v budoucnu opět komunikováno, zůstává objekt po vyprázdnění nadále zachován. Nevyužité dočasné paměti jsou periodicky každých 5 minut odstraňovány.

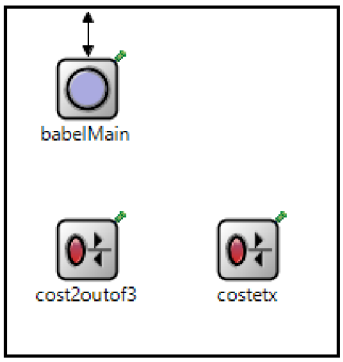


Obrázek 5.1: Zobrazení seznamu dočasných pamětí v prostředí OMNeT++/Tkenv

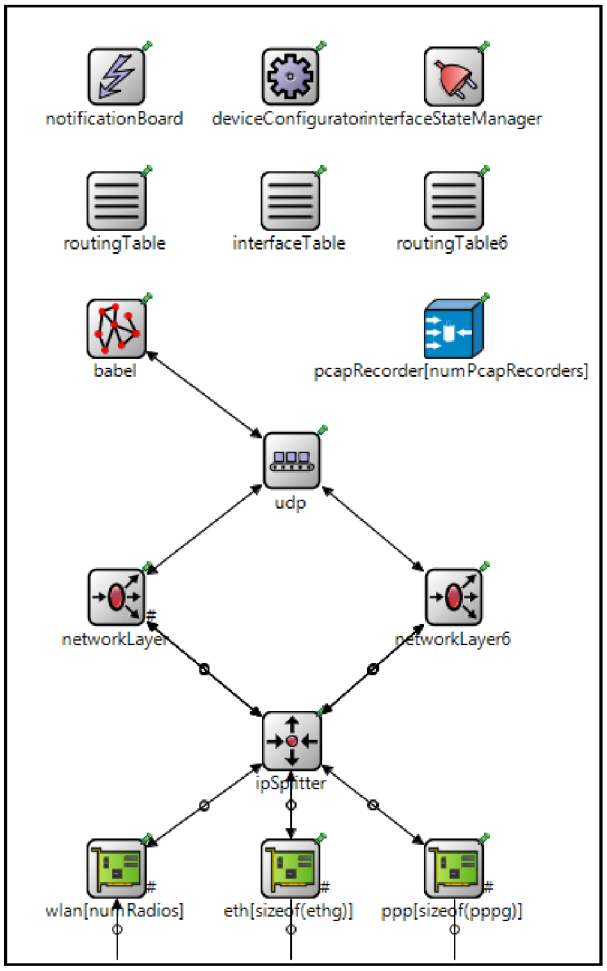
5.2 Modul BabelMain

Modul *BabelMain* implementuje samotnou funkcionalitu protokolu. Modelován je jako jednoduchý modul, jež je součástí složeného modulu *BabelRouting*. Struktura *BabelRouting* je zobrazena na obrázku 5.2. Pro připojení k transportní vrstvě je vybaven dvojicí bran, jednou vstupní, druhou výstupní. V programu je reprezentován třídou *BabelMain*. Ta zahrnuje všechny datové struktury potřebné pro běh protokolu, a to identifikátor směrovače, sekvenční číslo směrovače, tabulku rozhraní, tabulku sousedů, tabulku topologie, tabulku zdrojů, tabulku nevyřízených požadavků, tabulku nepotvrzených zpráv a dočasné paměti. Obsahuje také číslo UDP portu používaného pro komunikaci, a objekty pro příjem multicast provozu skupin ff02::1:6 a 224.0.0.111. Dále si udržuje přístup k seznamu fyzických rozhraní směrovače a ke směrovacím tabulkám protokolů IPv4 a IPv6. Je zaregistrován v modulu *NotificationBoard* pro odběr upozornění na změny stavu síťových rozhraní. Pro konfiguraci směrovače používá *DeviceConfigurator*, který byl rozšířen o podporu načítání konfigurace protokolu Babel.

Tento modul obstarává práci s časovači. Vytváří je, nastavuje a zpracovává jejich přijetí. Dále disponuje metodami pro odesílání a zpracování síťové komunikace, správu dočasných pamětí, práci s tabulkou nepotvrzených zpráv, výběr nejlepších cest a jejich instalaci do směrovacích tabulek.



Obrázek 5.2: Struktura složeného modulu BabelRouting



Obrázek 5.3: Struktura složeného modulu BabelRouter

5.3 Moduly výpočtu ceny linky

Výpočet ceny linky je obstaráván samostatnými moduly. Tento přístup umožňuje jejich snadné přidávání a úpravu. Počet těchto modulů není omezen. Který modul bude pro výpočet použit, je možné nastavit zvlášť pro každé síťové rozhraní, volbou *CostCompModule* v konfiguračním souboru.

Vyžadované metody jsou předepsány abstraktní třídou *IBabelCostComputation*, a to jmenovitě:

- *computeRxcost* - přijímající jako parametry vektor historie příjmu Hello zpráv a hodnotu RXCOST za ideálních podmínek (tzv. nominální RXCOST),
- *computeCost* - přijímající navíc hodnotu TXCOST.

Vyžaduje se reprezentace vektoru historie pomocí datového typu *uint16_t*, kdy nejvýznamnější bit nese informaci o nejčerstvějším Hello intervalu, tak jak je znázorněno na obrázku 5.5, ve kterém hodnota „1“ značí úspěšné přijetí Hello zprávy a „0“ nepřijetí.

Podle ukázkové implementace *babeld* jsou implementovány dvě základní strategie výpočtu, *k-out-of-j* a *ETX*.

Modul CostKoutofj

Je parametrizovatelný pro proměnné *k* a *j*. Při splnění podmínky, že v posledních *j* intervalech bylo úspěšně přijato alespoň *k* Hello zpráv vrací metoda *computeRxcost* hodnotu nominální RXCOST, v opačné případě *nekonečno*. Výpočet ceny linky pomocí *computeCost* odpovídá vztahu 2.3.

Modul CostEtx

Při výpočtu RXCOST je historie příjmu ohodnocena, a to jako suma hodnocení jednotlivých intervalů. Každý z třech posledních intervalů je v případě úspěšného přijetí hodnocen hodnotou 8192, v opačném případě hodnotou 0. Ohodnocení ostatních intervalů odpovídá hodnotě daného bitu v šestnáctibitovém datovém typu *uint16_t*. Samotná hodnota RXCOST je pak určena jako

$$RXCOST = \frac{32768}{RatedHistory + 1} * NominalRXCOST \tag{5.3}$$

Výpočet ceny linky pomocí *computeCost* odpovídá vztahu 2.7.

5.4 Datové struktury

Tato podkapitola nastiňuje způsob realizace významných datových struktur, použitých při implementaci funkcionality protokolu Babel.

5.4.1 Tabulka síťových rozhraní

Tabulka *BabelInterfaceTable* slouží pro uchování seznamu síťových rozhraní, která jsou zahrnuta do směrovacího procesu. Třída implementuje metody pro operace vkládání, vyhledávání a mazání záznamů. Položky jsou vkládány během inicializace *BabelMain* při načítání konfigurace z XML souboru pomocí modulu *deviceConfigurator*. Při vytváření je generováno náhodné sekvenční číslo. Jestliže je síťové rozhraní zapnuté, je odpovídající položka aktivována. Při aktivaci jsou dle konfigurace zaregistrovány objekty pro komunikaci pomocí UDP a náhodně načasovány časovače. Dále je krátce po aktivaci naplánováno odeslání Hello a Wildcard Route Request TLV, čímž je urychlena detekce sousedních směrovačů a získání jejich směrovacích informací. Existující položky mohou být aktivovány, případně deaktivovány, v reakci na změnu stavu síťového rozhraní během provádění simulace. Změna stavu je indikována výpisem v EvenLogu.

Samotné rozhraní je v tabulce reprezentováno třídou *BabelInterface*. Ta obsahuje konfigurační informace (typ a stav rozhraní, protokol používaný pro komunikaci, intervaly Hello a Update, nominální hodnotu RXCOST a stav funkce Split-Horizon), aktuální sekvenční číslo odesílané v Hello zprávách zasílaných daným rozhraním a ukazatele na časovače Hello a Update. Dále pak ukazatel na modul výpočtu ceny linky, díky čemuž může každé rozhraní používat rozdílný způsob hodnocení linky. Jelikož knihovna INET neumožňuje získání informací o připojených IPv6 prefixech v pozdější fázi běhu simulace, jsou uloženy i ty. Také jsou poznamenány datové struktury pro komunikaci pomocí protokolu UDP, a to samostatně pro IPv4 a IPv6, dle nastavení rozhraní. Implementovány jsou metody pro přístup k datovým položkám a pro práci se sekvenčním číslem a časovači.

Ve výchozím nastavení jsou intervaly Hello a Update určeny dle vztahů 5.4 a 5.5.

$$HelloInterval = \begin{cases} 4\,s & pro\ bezdrátová\ rozhraní \\ 20\,s & pro\ drátová\ rozhraní \end{cases} \tag{5.4}$$

$$UpdateInterval = 4 * HelloInterval \tag{5.5}$$

```
bit.getInterfaces() (std::vector<BabelInterface *>)
bit.getInterfaces()[5] (BabelInterface *)
    [0] = eth0:ena Send:NONE Dist:IPv6 SH:ena Wired:ena HSeqno:17089 HInt:2000 UInt:8000
    [1] = eth1:ena Send:IPv4 Dist:IPvX SH:ena Wired:ena HSeqno:32231 HInt:2000 UInt:8000
    [2] = eth2:ena Send:IPv6 Dist:IPv6 SH:dis Wired:ena HSeqno:24153 HInt:100 UInt:400
    [3] = eth3:ena Send:IPv6 Dist:NONE SH:dis Wired:ena HSeqno:27470 HInt:1100 UInt:4400
    [4] = wlan0:ena Send:IPv6 Dist:IPv6 SH:dis Wired:dis HSeqno:2164 HInt:400 UInt:1600
```

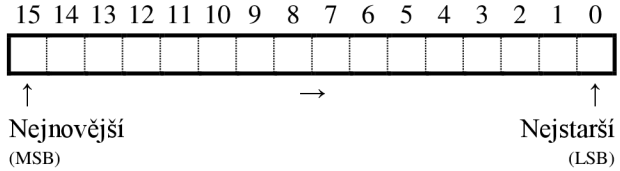
Obrázek 5.4: Zobrazení tabulky síťových rozhraní v prostředí OMNeT++/Tkenv

5.4.2 Tabulka sousedních směrovačů

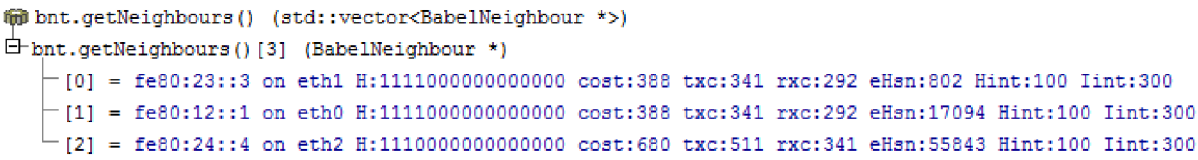
Datová struktura *BabelNeighbourTable* uchovává informace o sousedních směrovačích, se kterými byl navázán vztah sousedství. Pro jeho navázání postačuje příjem Hello TLV. Třídou jsou implementovány metody pro operace přidávání, odebírání, vyhledávání a určení počtu sousedů na daném rozhraní. Položky jsou vytvářeny a mazány dynamicky za běhu, při získání či ztrátě vztahu sousedství. Vztah sousedství nastává při přijetí Hello zprávy od souseda, pro kterého dosud neexistoval záznam. Zaniká, pokud není v historii uvedeno žádné úspěšné přijetí Hello. O změně vztahu je informováno výpisem v EvenLogu.

Pro reprezentaci sousedního směrovače slouží třída *BabelNeighbour*. Ta zahrnuje informace o jeho adrese síťové vrstvy, lokálním rozhraní, na kterém je soused dostupný, historii příjmu sousedem odeslaných Hello zpráv, hodnocení příjmu Hello zpráv z pohledu souseda, ceně linky, očekávaném sekvenčním čísle následujícího Hello TLV, hodnotách intervalů Hello a IHU, a časovačích pro určení vypršení doby těchto intervalů. Adresa síťové vrstvy je uchovávána pomocí univerzálního datového typu *IPvXAddress*, jež je schopen pojmout jak IPv4, tak i IPv6 adresu. To umožňuje, aby tabulka sousedů pojala všechny sousední směrovače bez rozlišování protokolu síťové vrstvy, kterým komunikují. Třída disponuje metodami pro přístup k datovým položkám, započítávání příjmu či ztráty Hello TLV do historie, přepočítání ceny linky a RXCOST, a správu časovačů.

Pro ukládání historie příjmu je využit datový typ *uint16_t* o velikosti 16 bitů. Informace o jedné zprávě je reprezentována jedním bitem, kdy hodnota „1“ znamená úspěšné přijetí, a „0“ nedoručení. Na historii je pohlíženo jako na vektor položek nesoucí informaci pro posledních 16 intervalů. Nejčerstvější položka je uložena na pozici nejvýznamnějšího bitu. Při vkládání jsou existující položky posouvány o jednu pozici doprava.



Obrázek 5.5: Vektor reprezentující historii příjmu zpráv



Obrázek 5.6: Zobrazení tabulky sousedních směrovačů v prostředí OMNeT++/Tkenv

5.4.3 Tabulka topologie

Pro reprezentaci tabulky topologie slouží třída *BabelTopologyTable*, uchovávající seznam všech cest, které jsou směrovači známé. To jsou buď cesty k přímo připojeným sítím, nebo cesty získané od sousedních směrovačů výměnou zpráv. Kromě základních metod pro operace jako vkládání, mazání a vyhledávání podle indexu jsou třídou implementovány i metody specifické. Například vyhledání cesty nepoužívající uvedený následující skok, či dotaz, zda existuje v tabulce cesta s cílovým prefixem kratším a zároveň zahrnujícím uvedený prefix. Záznamy jsou do tabulky vkládány, respektive z ní odstraňovány, dynamicky během simulace, jako reakce na získání nových, či ztrátu stávajících cest. Při změně parametrů jsou aktualizovány. O vytvoření nové položky je uveden záznam v EventLogu.

Položky tabulky topologie jsou instancemi třídy *BabelRoute*. I přesto, že knihovna INET nabízí předpřipravené třídy pro reprezentaci směrovacích cest, nebyly pro tvorbu *BabelRoute* využity, kvůli specifickým nárokům - především nezávislost na síťovém protokolu. Obsaženy jsou informace o prefixu cílové sítě, identifikátoru směrovače, od nějž informace pochází. Taktéž je poznamenán soused, který informaci předal, s jakou vzdáleností a přes který následující skok je dostupná. Pro správu cest jsou však nutné další položky. Jsou to příznak, značící výběr cesty jako nejlepší možné, interval, jak často jsou pro danou cestu zasílány pravidelné aktualizace, ukazatel do směrovací tabulky v případě, že je cesta používána pro směrování, a časovač vypršení platnosti cesty.

Protože se protokol Babel snaží aktivně předcházet vypršení platnosti cesty, je těsně před expirací cesty odeslán požadavek na informaci o dané cestě sousedovi, od kterého tuto cestu získal, a to v podobě RouteRequest TLV. V případě existence cesty je doručena aktualizace, a doba platnosti prodloužena. V případě neexistence, soused reaguje odvoláním cesty, a ta je ihned zneplatněna. Pro implementaci tohoto mechanismu je využit časovač, nastavovaný na dobu

$$BefRouteExpiryInterval = \frac{7}{8} * RouteExpiryInterval, \tag{5.6}$$

kde

$$RouteExpiryInterval = 3{,}5 * UpdateInterval. \tag{5.7}$$

```
btt.getRoutes() (std::vector<BabelRoute *>)
btt.getRoutes()[13] (BabelRoute *)
  [0] = > 2001:db8:a::/64 local  metric:0 orig:1111:1111:1111:1111
  [1] = > 2001:db8:12::/64 local  metric:0 orig:1111:1111:1111:1111
  [2] = > 2001:db8:13::/64 local  metric:0 orig:1111:1111:1111:1111
  [3] = > 2001:db8:c::/64 NH:fe80:13::3 metric:256 orig:3333:3333:3333:3333 from:fe80:13::3 RD:(31921, 0), in RT
  [4] = > 2001:db8:23::/64 NH:fe80:13::3 metric:256 orig:3333:3333:3333:3333 from:fe80:13::3 RD:(31921, 0), in RT
  [5] =   2001:db8:13::/64 NH:fe80:13::3 metric:256 orig:3333:3333:3333:3333 from:fe80:13::3 RD:(31921, 0)
  [6] = > 2001:db8:b::/64 NH:fe80:12::2 metric:256 orig:2222:2222:2222:2222 from:fe80:12::2 RD:(27469, 0), in RT
  [7] =   2001:db8:12::/64 NH:fe80:12::2 metric:256 orig:2222:2222:2222:2222 from:fe80:12::2 RD:(27469, 0)
  [8] =   2001:db8:23::/64 NH:fe80:12::2 metric:256 orig:2222:2222:2222:2222 from:fe80:12::2 RD:(27469, 0)
  [9] = > 2001:db8:24::/64 NH:fe80:12::2 metric:256 orig:2222:2222:2222:2222 from:fe80:12::2 RD:(27469, 0), in RT
  [10] =   2001:db8:13::/64 NH:fe80:12::2 metric:512 orig:3333:3333:3333:3333 from:fe80:12::2 RD:(31921, 256)
  [11] =   2001:db8:12::/64 NH:fe80:13::3 metric:512 orig:2222:2222:2222:2222 from:fe80:13::3 RD:(27469, 256)
  [12] = > 2001:db8:d::/64 NH:fe80:12::2 metric:512 orig:4444:4444:4444:4444 from:fe80:12::2 RD:(53887, 256), in RT
```

Obrázek 5.7: Zobrazení tabulky topologie v prostředí OMNeT++/Tkenv

5.4.4 Tabulka zdrojů

Pro uchovávání tzv. vhodných vzdálenost slouží tabulka zdrojů. Je implementována třídou *BabelSourceTable*, která poskytuje operace pro přidávání, vyhledávání a mazání. Zdroje jsou instancemi třídy *BabelSource*, která obsahuje prefix cílové sítě, identifikátor směrovače, od nějž informace pochází a vhodnou vzdálenost, nazývanou FD. Zdroje se vytvářejí, případně aktualizují, vždy při odesílání aktualizací sousedním směrovačům. Jejich platnost je omezena na 180 sekund bez aktualizace, po jejichž uplynutí jsou vymazány.

S využitím informací z této datové struktury jsou odfiltrovány příchozí aktualizace cest, jež by potenciálně mohly způsobit vytvoření směrovací smyčky v topologii. Toho je docíleno porovnáním přijaté vzdálenosti s vlastní ohlašovanou. Je-li striktně menší je aktualizace akceptována. Přijata je i v případě, kdy se jedná o odvolání cesty (nekonečná metrika), nebo pro danou cestu neexistuje v tabulce zdrojů záznam.

```
bst.getSources() (std::vector<BabelSource *>)
bst.getSources()[8] (BabelSource *)
  [0] = 2001:db8:b::/64 orig:2222:2222:2222:2222 FD:(27469, 0)
  [1] = 2001:db8:12::/64 orig:2222:2222:2222:2222 FD:(27469, 0)
  [2] = 2001:db8:23::/64 orig:2222:2222:2222:2222 FD:(27469, 0)
  [3] = 2001:db8:24::/64 orig:2222:2222:2222:2222 FD:(27469, 0)
  [4] = 2001:db8:c::/64 orig:3333:3333:3333:3333 FD:(31921, 680)
  [5] = 2001:db8:13::/64 orig:3333:3333:3333:3333 FD:(31921, 680)
  [6] = 2001:db8:a::/64 orig:1111:1111:1111:1111 FD:(52620, 680)
  [7] = 2001:db8:d::/64 orig:4444:4444:4444:4444 FD:(53887, 2042)
```

Obrázek 5.8: Zobrazení tabulky zdrojů v prostředí OMNeT++/Tkenv

5.4.5 Tabulka nevyřízených požadavků

Datová struktura uchovávající čekající požadavky na zvýšení sekvenčního čísla je reprezentována třídou s názvem *BabelPenSRTable*. Umožňuje základní operace, a to vkládání, vyhledávání a mazání. Pro reprezentaci položek slouží třída *BabelPenSR*, uchovávající nekomprimovaný požadavek Seqno Request FTLV a sousední směrovač od kterého byl požadavek přijat, v případě, že je požadavek přeposlán. Dále pak počet zbývajících pokusů na znovuodeslání, rozhraní a adresu cíle, na kterou má být případně požadavek opakovaně odeslán. Interval opakovaného odeslání je stanoven dle rovnice 5.8, počet pokusů je pevně stanoven na 3. Lokálně vzniklé Seqno Request TLV jsou odesílány s položkou Hop Count nastavenou na hodnotu 127.

$$SRResendInterval = \frac{HelloInterval}{2} \tag{5.8}$$

S použitím obsahu této struktury, je při přijetí aktualizace určováno, zda se jedná o reakci na přeposlaný požadavek Seqno Request TLV. Pokud ano, je bez odkladu odeslána aktualizace. O této události je vytvořen záznam v EventLogu.

```
bpsrt.getRequests() (std::vector<BabelPenSR *>)
bpsrt.getRequests()[4] (BabelPenSR *)
    [0] = 2001:db8:a::/64 orig:1111:1111:1111:1111 reqSN:52621 from:fe80:23::2 remains:3 on eth0
    [1] = 2001:db8:b::/64 orig:2222:2222:2222:2222 reqSN:27470 from:fe80:13::1 remains:3 on eth1
    [2] = 2001:db8:24::/64 orig:2222:2222:2222:2222 reqSN:27470 from:fe80:13::1 remains:3 on eth1
    [3] = 2001:db8:d::/64 orig:4444:4444:4444:4444 reqSN:53888 from:fe80:13::1 remains:3 on eth1
```

Obrázek 5.9: Zobrazení tabulky nevyřízených požadavků v prostředí OMNeT++/Tkenv

5.4.6 Tabulka nepotvrzených zpráv

Protokolem Babel je v určitých případech požadováno spolehlivé zaslání zpráv. To je realizováno pomocí požadavku na zaslání potvrzení přijetí. V době mezi odesláním tohoto požadavku a přijetím všech potvrzení jsou informace o takovýchto zprávách uloženy v tabulce nepotvrzených zpráv. Její položky jsou instancemi třídy *BabelToAck*. Každá pak obsahuje náhodné číslo *nonce*, seznam adresátů, od nichž je vyžadováno přijetí potvrzení, počet zbývajících pokusů opětovného odeslání, cílovou adresu, odchozí rozhraní a duplikát samotné zprávy. Tato třída disponuje metodami pro přístup k datovým položkám a práci se seznamem adresátů. Interval opakovaného odeslání je určen shodně jako v případě opětovného odesílání požadavku na zvýšení sekvenčního čísla, jak je uvedeno ve vztahu 5.8. Počet pokusů je roven 3.

5.4.7 Pomocné datové struktury

Pro usnadnění jsou definovány pomocné datové struktury, zapouzdřující související položky do logického celku. Jejich využití ve složitějších strukturách značně zpřehledňuje návrh a usnadňuje práci s nimi.

BabelMessage

Třída představující samotný obsah zprávy zasílané mezi Babel směrovači. Automaticky vyplňuje položky hlavičky zprávy a přepočítává délku těla. Definuje metody pro přidávání dat do těla zprávy, průchod po jednotlivých TLV a započítávání statistických údajů. Taktéž umožňuje zobrazení přenášeného obsahu přímo v grafickém simulačním prostředí, a to ve stručné i detailní podobě, jak ilustruje obrázek 5.10.

```
info = 'HELLO, IHU, ROUTERID, UPDATE, UPDATE, UPDATE, ROUTERID, UPDATE, UPDATE' (string)
detailedInfo =
 'HELLO: Length=6, Seqno=17355, Interval=100
 IHU: Length=22, AE=2, Rxcost=256, Interval=300, Address=fe.80.00.12.00.00.00.00.00.00.00.00.00.00.00.02
 ROUTERID: Length=10, Router-id=1111:1111:1111:1111
 UPDATE: Length=18, AE=2, Flags=80, Plen=64, Omitted=0, Interval=400, Seqno=52620, Metric=0, Prefix=20.01.0d.b8.00.0a.00.00
 UPDATE: Length=13, AE=2, Flags=00, Plen=64, Omitted=5, Interval=400, Seqno=52620, Metric=0, Prefix=12.00.00
 UPDATE: Length=13, AE=2, Flags=00, Plen=64, Omitted=5, Interval=400, Seqno=52620, Metric=0, Prefix=13.00.00
 ROUTERID: Length=10, Router-id=3333:3333:3333:3333
 UPDATE: Length=13, AE=2, Flags=00, Plen=64, Omitted=5, Interval=400, Seqno=31921, Metric=256, Prefix=0c.00.00
 UPDATE: Length=13, AE=2, Flags=00, Plen=64, Omitted=5, Interval=400, Seqno=31921, Metric=256, Prefix=23.00.00'
```

Obrázek 5.10: Zobrazení obsahu přenášené zprávy v prostředí OMNeT++/Tkenv

routeDistance

Třída *routeDistance* reprezentuje vzdálenost cesty. Zapouzdřuje sekvenční číslo a metriku. Pro intuitivní používání při porovnávání, jsou definovány operátory ==, !=, <, > <= a >=.

netPrefix

Tato třída představuje prefix sítě reprezentovaný jako adresa a délka. Pro možnost jejího obecného použití je realizována jako šablona, umožňující instanciaci jak s využitím univerzálního typu *IPvXAddress*, tak i specifických typů *IPv4Address* a *IPv6Address*. Disponuje metodami pro komprimaci a dekomprimaci.

rid

Osmibajtový identifikátor směrovače. Dovoluje přímé načítání z objektu *IPv6Address* a vzájemné porovnání.

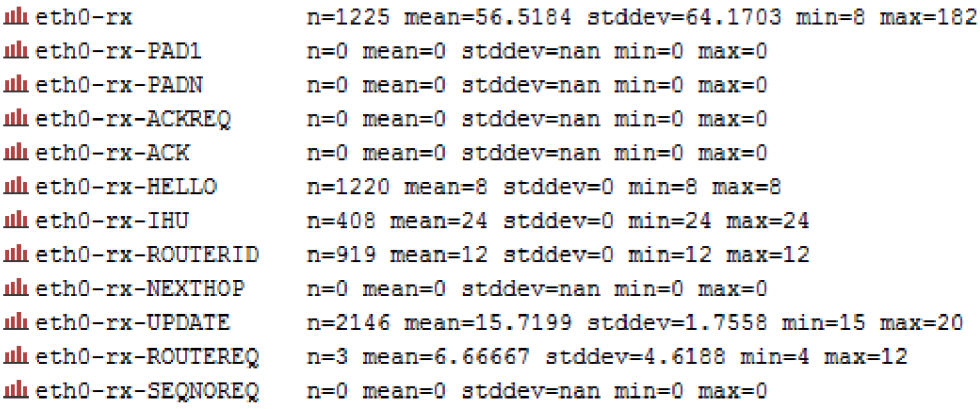
BabelStats

Datová struktura zapouzdřující objekty statistických údajů pro celé zprávy i jednotlivá TLV. Navržena pro uchování informací o komunikaci v jednom směru.

5.5 Výstup simulace

Kromě samotné funkcionality protokolu, kterou je v simulačním prostředí možné krok po kroku detailně zkoumat, umožňuje simulační model oproti reálné implementaci navíc sběr podrobných informací o komunikaci mezi směrovači. Ty jsou zaznamenávány na rozhraních v odchozím i příchozím směru. Obsahují informace o počtu a velikosti jak celých zpráv, tak jednotlivých typů TLV. Pro reprezentaci těchto údajů je využita třída *cStdDev*, umožňující přístup k jejímu obsahu i pří průběhu simulace přímo z grafického rozhraní, jak je vidět na obrázku 5.11. Poskytuje metody pro práci s údaji, zahrnující určení počtu, minima, maxima, průměru a dalších statistických veličin. Po skončení simulace jsou statistické údaje ve stručném formátu vypsány do EvenLogu, a v plné formě uloženy do souboru obsahující výsledky simulace.

Tyto výstupy mohou být užitečné pro srovnání protokolu Babel s jinými směrovacími protokoly z pohledu nároků na komunikaci. Také mohou pomoci pro nalezení optimální konfigurace, či při analýze úspěšnosti komprese zpráv.



Obrázek 5.11: Zobrazení statistických dat o zprávách přijatých rozhraním eth0 v prostředí OMNeT++/Tkenv

6 Porovnání implementací

K ověření správnosti vytvořeného modelu, je provedeno porovnání chování a výsledků v simulačním a reálném prostředí. Pro tyto účely je používán program *babeld* ve verzi 1.5.1, spuštěný na systému s OS Debian 7.7 x64 (3.2.65-1). Síťová komunikace mezi jednotlivými systémy byla zachytávána a analyzována programem Wireshark.

Při porovnávání zasílaných zpráv je nutné brát v úvahu složitý mechanismus optimalizace s použitím dočasného pozdržení jednotlivých TLV, jehož doba je navíc náhodně generovaná.

6.1 Navázání vztahu sousedství

V prvním testu je předveden proces navázání vztahu sousedství mezi směrovači. Je proveden za situace, kdy směrovač R2 je v provozu, a R1 se v čase T_0 připojí do sítě. Toho je v reálném prostředí dosaženo spuštěním procesu *babeld*. V simulaci je pomocí modulu *scenarioManager* aktivováno síťové rozhraní.

Babeld implementuje pokročilé postupy pro zvýšení spolehlivosti při přenosu po ztrátových linkách, které v simulačním modelu realizovány nejsou. Z tohoto důvodu jsou zprávy zasílané vícenásobně při porovnávání ignorovány.

Pořadí	TLV nesené ve zprávě	Směr přenosu	Čas v simulaci [s]	Čas v realitě [s]
1	Hello, Route Request	R1→R2	0,092	0,006
2	Hello, IHU, Update	R2→R1	0,292	0,007
3	Hello, IHU	R1→R2	0,492	0,040
4	Hello, IHU	R2→R1	0,692	0,134
5	Route Request	R2→R1 (unicast)	0,692	0,903
6	Hello, IHU (Update)	R1→R2	0,892	1,084
7	Route Request	R1→R2 (unicast)	0,892	1,085
8	Update (IHU)	R2→R1	1,902	1,744
9	Hello, IHU	R2→R1	5,632	5,111

Tabulka 1: Porovnání zasílaných zpráv při navazování vztahu sousedství

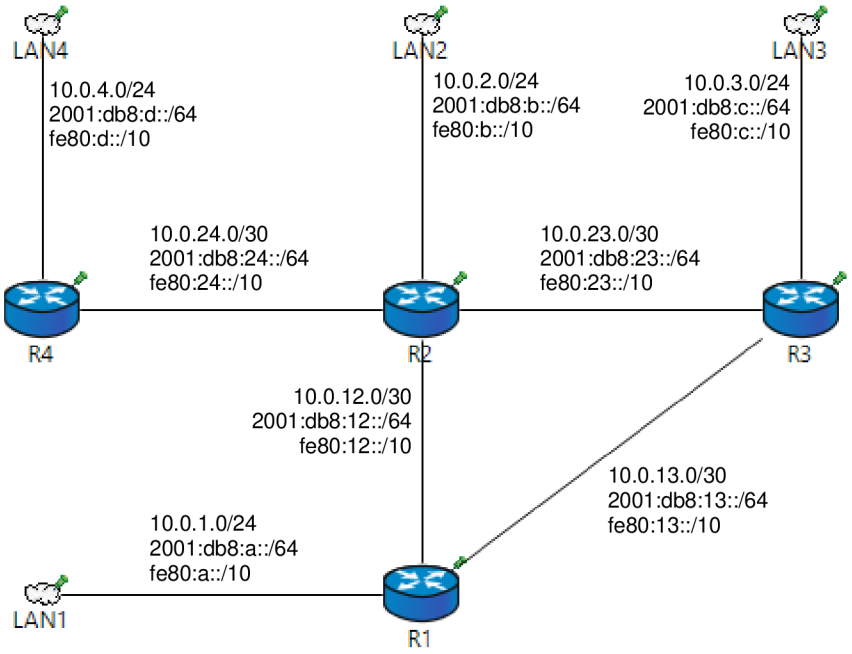
V tabulce Tabulka 1 jsou uvedeny zprávy zasílané mezi směrovači při ustanovení vztahu sousedství. První zpráva je zasílána nově připojeným směrovačem, který se pomocí ní ohlašuje ostatním a žádá je o zaslání směrovacích informací. V reakci na ni se nově připojenému směrovači soused taktéž ohlásí, a připojí informace o jemu známých cestách. To je z pohledu R1 první Hello TLV doručené od sousedního směrovače. Z tohoto důvodu na ni odpovídá třetí zprávou v pořadí, nesoucí Hello a IHU TLV. Na tomto místě je vhodné poznamenat, že komunikace probíhá po drátové lince, na

které je pro hodnocení její kvality používána strategie *2-out-of-3*, a proto je přijetí druhého Hello pro ohodnocení linky zlomové, neboť dosud byla linka považována za nedostupnou. Při přijetí této zprávy je z pohledu R2 doručeno druhé Hello, na což reaguje opět zasláním Hello a IHU TLV (zpráva č. 4). Ovšem navíc je odeslán i požadavek na zaslání všech známých cest, adresovaný pouze danému sousedovi, v tabulce označen číslem 5. Na doručení čtvrté zprávy reaguje R1 stejným způsobem jako R2 na třetí zprávu. V důsledku reakce na Route Request TLV je však do šesté zprávy kromě Hello a IHU zahrnuto i Update TLV, a to díky mechanismu dočasnému pozdržení odesílaných zpráv. Přijetím zprávy č. 6 je doručeno od R1 již třetí Hello, na které už není důvod reagovat stejným způsobem jako na první dvě. Osmá zpráva nese informace o cestách jako odpověď na požadavek ze zprávy sedmé. Vypršení časovače Hello, který je v daném příkladu nastaven na 4 s, má za následek odeslání deváté zprávy.

Při porovnání byly odhaleny odlišnosti v nesených TLV. TLV navíc jsou ve druhém sloupci tabulky uvedeny v závorce. První nekonzistence je způsobena již zmíněným rozdílem implementací ve snaze zvýšení spolehlivosti na ztrátových linkách. Díky tomu (a také díky mechanismu dočasného pozdržení) bylo Update TLV ze šesté zprávy přeneseno ve zprávě jiné (v tabulce neuvedené). Stejnému důvodu přikládám také za následek časový rozdíl u čtvrté zprávy, jelikož byla stejná dočasná paměť využívána jinými TLV, což pravděpodobně ovlivnilo dobu dočasného uložení. Rozdíl ve zprávě č. 8 je způsoben odlišným nakládáním s požadavkem na zaslání informací o všech směrovači známých cestách. Zatímco implementovaný model na tento požadavek odpovídá vždy, *babeld* jej považuje za možnou hrozbu, a často zasílané požadavky ignoruje.

6.2 Výsledná topologie

Druhý test se zaměřuje na srovnání obsahu topologických tabulek jednotlivých směrovačů, neboť ty odpovídají výsledné topologii bez smyček, jejíž určení je ostatně hlavním cílem směrovacích protokolů.



Obrázek 6.1: Topologie použitá při testování

Test byl proveden na topologii ilustrované obrázkem Obrázek 6.1. Všechny linky mezi směrovači jsou drátové, určující cenu linky strategií 2-out-of-3, s Hello intervalem nastaveným na 4 s, a aktivním Split-Horizon. Pro přenos zpráv je používán protokol IPv6. Propagovány jsou pouze IPv6 sítě. Program *babeld* byl spuštěn příkazem:

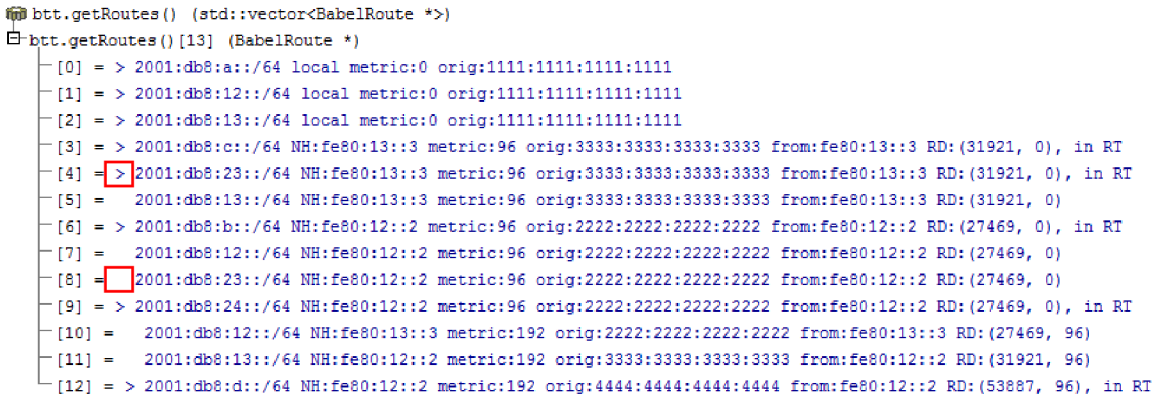
```
babeld  -C 'redistribute ip ::/0 le 64' \
        -C 'redistribute local deny' \
        eth0 [eth1] [eth2]
```

Obsah těchto tabulek byl zaznamenán po dostatečně dlouhé době pro konvergenci sítě. I přesto, že linkové-lokálních IPv6 adresy se v simulačním a reálném prostředí liší, je v obou případech pro snadnější orientaci dodržováno pravidlo, kdy poslední bajt adresy se shoduje s číslem v názvu směrovače.

Porovnání TT směrovače R1

Při porovnávání topologických tabulek mezi simulačním modelem a programem *babeld* byly u směrovače R1 vypozorovány odlišnosti, které jsou znázorněny v tabulce Tabulka 2 a obrázku Obrázek 6.2. Jak je patrné, do sítě *2001:db8:23::/64* zná směrovač v obou případech dvě cesty. První vedoucí přes směrovač R3, druhou přes směrovač R2. Obě tyto cesty jsou dle metriky rovnocenné, proto je zcela korektním chováním zvolit kteroukoli z nich. Jelikož protokol Babel nepodporuje vyrovnávání zátěže, a to ani na shodně hodnocených linkách, je nutné vždy vybrat právě jednu. Tato volba je ovlivněna zejména pořadím získání informací o cestách. První naučená cesta je ihned zvolena, a dokud je dostupná lze ji nahradit pouze lépe hodnocenou cestou.

Z předchozího popisu problému vyplývá, že není ovlivnitelný implementací. Jak bude demonstrováno dále v textu, tato odlišnost ve zvolené topologii ovlivní tabulky topologií ostatních směrovačů.



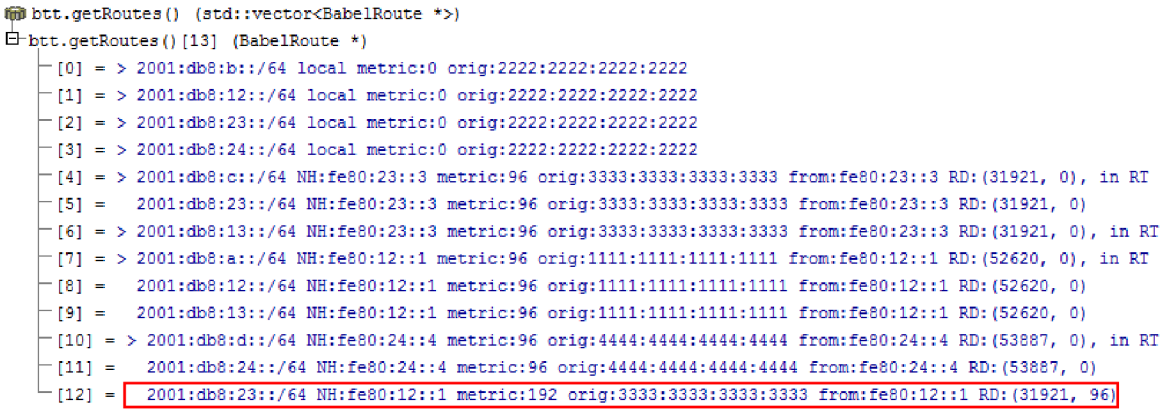
Obrázek 6.2: Tabulka topologie směrovače R1 v simulačním prostředí, s červeně zvýrazněnými odlišnostmi oproti testu v reálné prostředí

Prefix	Metrika	Ohlášená metrika	RouterID původce	Sekvenční číslo	Odchozí rozhraní	Následující skok	Příznak
2001:db8:a::/64	0						(exported)
2001:db8:12::/64	0						(exported)
2001:db8:13::/64	0						(exported)
2001:db8:b::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:1202	(installed)
2001:db8:c::/64	96	0	0a:00:00:ff:fe:00:13:03	46989	eth1	fe80::a00:ff:fe00:1303	(installed)
2001:db8:d::/64	192	96	0a:00:00:ff:fe:00:24:04	4932	eth0	fe80::a00:ff:fe00:1202	(installed)
2001:db8:12::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:1202	(feasible)
2001:db8:12::/64	192	96	0a:00:00:ff:fe:00:12:02	26430	eth1	fe80::a00:ff:fe00:1303	(feasible)
2001:db8:13::/64	192	96	0a:00:00:ff:fe:00:13:03	46989	eth0	fe80::a00:ff:fe00:1202	(feasible)
2001:db8:13::/64	96	0	0a:00:00:ff:fe:00:13:03	46989	eth1	fe80::a00:ff:fe00:1303	(feasible)
2001:db8:23::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:1202	(installed)
2001:db8:23::/64	96	0	0a:00:00:ff:fe:00:13:03	46989	eth1	fe80::a00:ff:fe00:1303	(feasible)
2001:db8:24::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:1202	(installed)

Tabulka 2: Tabulka topologie směrovače R1 v reálném prostředí, s červeně zvýrazněnými odlišnostmi oproti testu v simulačním prostředí

Porovnání TT směrovače R2

Díky rozdílné volbě cesty směrovačem R1 jsou jím propagované cesty pochopitelně taktéž rozdílné. To je pozorovatelné na R2. V simulačním prostředí získává R2 informaci o exitující cestě do sítě *2001:db8:23::/64* propagované směrovačem R3, přes R1. V reálném prostředí, kdy R1 zvolil cestu vedoucí přes R2, tuto informaci směrovač R1 zpět R2 nezasílá díky aktivní funkcionalitě Split-Horizon. V případě vypnutí Split-Horizon by však taková informace nesplňovala podmínku vhodnosti.



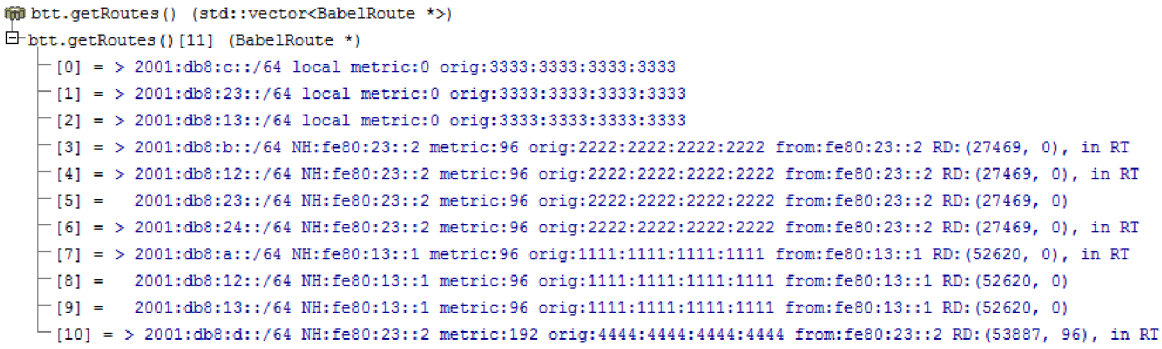
Obrázek 6.3: Tabulka topologie směrovače R2 v simulačním prostředí, s červeně zvýrazněnou přebývající cestou

Prefix	Metrika	Ohlášená metrika	RouterID původce	Sekvenční číslo	Odchozí rozhraní	Následující skok	Příznak
2001:db8:b::/64	0						(exported)
2001:db8:12::/64	0						(exported)
2001:db8:23::/64	0						(exported)
2001:db8:24::/64	0						(exported)
2001:db8:a::/64	96	0	0a:00:00:ff:fe:00:12:01	36149	eth0	fe80::a00:ff:fe00:1201	(installed)
2001:db8:c::/64	96	0	0a:00:00:ff:fe:00:13:03	46989	eth3	fe80::a00:ff:fe00:2303	(installed)
2001:db8:d::/64	96	0	0a:00:00:ff:fe:00:24:04	4932	eth2	fe80::a00:ff:fe00:2404	(installed)
2001:db8:12::/64	96	0	0a:00:00:ff:fe:00:12:01	36149	eth0	fe80::a00:ff:fe00:1201	(feasible)
2001:db8:13::/64	96	0	0a:00:00:ff:fe:00:13:03	46989	eth3	fe80::a00:ff:fe00:2303	(installed)
2001:db8:13::/64	96	0	0a:00:00:ff:fe:00:12:01	36149	eth0	fe80::a00:ff:fe00:1201	(feasible)
2001:db8:23::/64	96	0	0a:00:00:ff:fe:00:13:03	46989	eth3	fe80::a00:ff:fe00:2303	(feasible)
2001:db8:24::/64	96	0	0a:00:00:ff:fe:00:24:04	4932	eth2	fe80::a00:ff:fe00:2404	(feasible)

Tabulka 3: Tabulka topologie směrovače R2 v reálném prostředí

Porovnání TT směrovače R3

Na směrovači R3 je rovněž patrný vliv rozhodnutí směrovače R1. Na rozdíl od R2 se však dle očekávání projeví opačně – směrovací informací navíc disponuje směrovač v reálném prostředí.



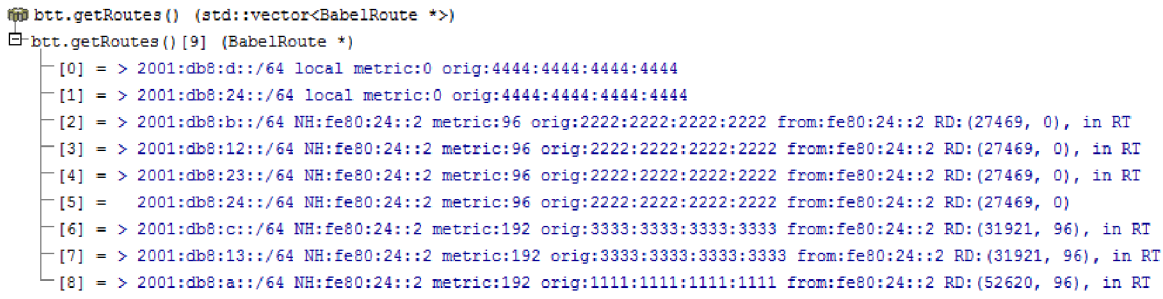
Obrázek 6.4: Tabulka topologie směrovače R3 v simulačním prostředí

Prefix	Metrika	Ohlášená metrika	RouterID původce	Sekvenční číslo	Odchozí rozhraní	Následující skok	Příznak
2001:db8:c::/64	0						(exported)
2001:db8:13::/64	0						(exported)
2001:db8:23::/64	0						(exported)
2001:db8:a::/64	96	0	0a:00:00:ff:fe:00:12:01	36149	eth0	fe80::a00:ff:fe00:1301	(installed)
2001:db8:b::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth1	fe80::a00:ff:fe00:2302	(installed)
2001:db8:d::/64	192	96	0a:00:00:ff:fe:00:24:04	4932	eth1	fe80::a00:ff:fe00:2302	(installed)
2001:db8:12::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth1	fe80::a00:ff:fe00:2302	(installed)
2001:db8:12::/64	96	0	0a:00:00:ff:fe:00:12:01	36149	eth0	fe80::a00:ff:fe00:1301	(feasible)
2001:db8:13::/64	96	0	0a:00:00:ff:fe:00:12:01	36149	eth0	fe80::a00:ff:fe00:1301	(feasible)
2001:db8:23::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth1	fe80::a00:ff:fe00:2302	(feasible)
2001:db8:23::/64	192	96	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:1301	(feasible)
2001:db8:24::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth1	fe80::a00:ff:fe00:2302	(installed)

Tabulka 4: Tabulka topologie směrovače R3 v reálném prostředí, s červeně zvýrazněnou přebývající cestou

Porovnání TT směrovače R4

Jelikož je v testovací topologii směrovač R4 připojen ke zbytku sítě pouze jednou linkou k R2, není výše uvedenou anomálií nijak zasažen. Tabulky topologií jsou mezi simulačním a reálným prostředím zcela shodné, jak je možné určit z obrázku Obrázek 6.5 a tabulky Tabulka 5.



Obrázek 6.5: Tabulka topologie směrovače R4 v simulačním prostředí

Prefix	Metrika	Ohlášená metrika	RouterID původce	Sekvenční číslo	Odchozí rozhraní	Následující skok	Příznak
2001:db8:d::/64	0						(exported)
2001:db8:24::/64	0						(exported)
2001:db8:a::/64	192	96	0a:00:00:ff:fe:00:12:01	36149	eth0	fe80::a00:ff:fe00:2402	(installed)
2001:db8:b::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:2402	(installed)
2001:db8:c::/64	192	96	0a:00:00:ff:fe:00:13:03	46989	eth0	fe80::a00:ff:fe00:2402	(installed)
2001:db8:12::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:2402	(installed)
2001:db8:13::/64	192	96	0a:00:00:ff:fe:00:13:03	46989	eth0	fe80::a00:ff:fe00:2402	(installed)
2001:db8:23::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:2402	(installed)
2001:db8:24::/64	96	0	0a:00:00:ff:fe:00:12:02	26430	eth0	fe80::a00:ff:fe00:2402	(feasible)

Tabulka 5: Tabulka topologie směrovače R4 v reálném prostředí

Při tomto testu byly odhaleny drobné rozdíly mezi simulačním a reálným prostředím v obsahu topologických tabulek. Jak již bylo popsáno, je způsoben vlivem pořadí získání informací, jež není možné ovlivnit. Navíc se dotýká pouze nepoužívaných cest, a tudíž na směrování datového provozu nemá žádný vliv.

6.3 Výpadek linky

Třetí test se zaměřuje na kontrolu chování modelu při ztrátě spojení mezi směrovači. Při testu jsou porovnány obsahy topologických tabulek jak před událostí, tak po ní. Detailně se zaměřuje na komunikaci krátce po výpadku, jejímž účelem je získání jiné cesty do cíle, pro nějž se aktuálně zvolená cesta stala nedostupnou. Scénář výpadku je proveden na topologii z obrázku Obrázek 6.1. V čase T_0 je spojení mezi R1 a R2 přerušeno. Toho je dosaženo v reálném prostředí pomocí odpojení kabelu, v simulaci pomocí komponenty scenarioManager, který ve stanovený čas deaktivuje síťová rozhraní směrovačů R1 a R2. Výpadek nastává v době, kdy je síť ve zkonvergovaném stavu.

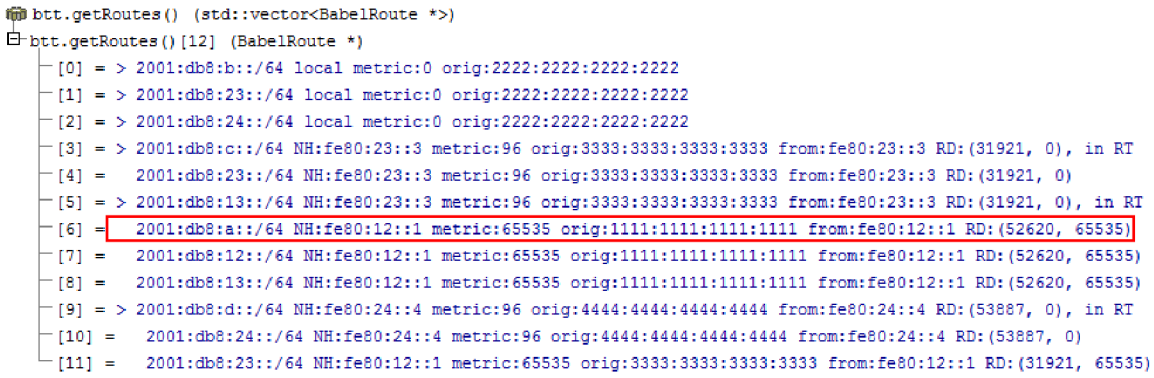
Jelikož je linka mezi R1 a R2 v dané topologii zvolená pro více cest, její výpadek vyvolá celou řadu reakcí. Pro názornost se věnujme pouze sledování stavu cesty do sítě 2001:db8:a::/64 ze směrovače R2.

```
btt.getRoutes() (std::vector<BabelRoute *>)
btt.getRoutes()[13] (BabelRoute *)
  [0] = > 2001:db8:b::/64 local metric:0 orig:2222:2222:2222:2222
  [1] = > 2001:db8:12::/64 local metric:0 orig:2222:2222:2222:2222
  [2] = > 2001:db8:23::/64 local metric:0 orig:2222:2222:2222:2222
  [3] = > 2001:db8:24::/64 local metric:0 orig:2222:2222:2222:2222
  [4] = > 2001:db8:c::/64 NH:fe80:23::3 metric:96 orig:3333:3333:3333:3333 from:fe80:23::3 RD:(31921, 0), in RT
  [5] =   2001:db8:23::/64 NH:fe80:23::3 metric:96 orig:3333:3333:3333:3333 from:fe80:23::3 RD:(31921, 0)
  [6] = > 2001:db8:13::/64 NH:fe80:23::3 metric:96 orig:3333:3333:3333:3333 from:fe80:23::3 RD:(31921, 0), in RT
  [7] = > 2001:db8:a::/64 NH:fe80:12::1 metric:96 orig:1111:1111:1111:1111 from:fe80:12::1 RD:(52620, 0), in RT
  [8] =   2001:db8:12::/64 NH:fe80:12::1 metric:96 orig:1111:1111:1111:1111 from:fe80:12::1 RD:(52620, 0)
  [9] =   2001:db8:13::/64 NH:fe80:12::1 metric:96 orig:1111:1111:1111:1111 from:fe80:12::1 RD:(52620, 0)
  [10] = > 2001:db8:d::/64 NH:fe80:24::4 metric:96 orig:4444:4444:4444:4444 from:fe80:24::4 RD:(53887, 0), in RT
  [11] =   2001:db8:24::/64 NH:fe80:24::4 metric:96 orig:4444:4444:4444:4444 from:fe80:24::4 RD:(53887, 0)
  [12] =   2001:db8:23::/64 NH:fe80:12::1 metric:192 orig:3333:3333:3333:3333 from:fe80:12::1 RD:(31921, 96)
```

Obrázek 6.6: Tabulka topologie směrovače R2 před výpadkem

Obrázek 6.6 zachycuje obsah tabulky topologie před výpadkem. Je z něj patrné, že zvolená cesta do sítě 2001:db8:a::/64 vede přes R1 s metrikou 96 a sekvenčním číslem 52620. Žádná záložní cesta pro tuto síť v tabulce neexistuje.

Jako reakce na výpadek jsou cesty dostupné přes ztracenou linku odvolány (metrika je nastavena na nekonečno). Tabulku topologie krátce po výpadku s cestami v tomto stavu zobrazuje obrázek Obrázek 6.7.



Obrázek 6.7: Tabulka topologie směrovače R2 krátce po výpadku

V případě, kdy by existovala záložní vhodná cesta, byla by ihned zvolena. To však pro síť 2001:db8:a::/64 neplatí. Jelikož z důvodu zajištění bezsmyčkové topologie nemůže být odstraněn záznam pro danou síť z tabulky zdrojů, byly by i nadále aktualizace od R3 propagující zmíněnou síť s metrikou 96 díky nesplnění podmínky vhodnosti ignorovány. Tím by na dobu platnosti záznamu zdroje došlo k uváznutí, během kterého by cílová síť nebyla dostupná. Jak je popisováno v kapitole 2.3.4, protokol Babel tuto situaci řeší požadavkem na zvýšení sekvenčního čísla. Tato komunikace je znázorněna v tabulce Tabulka 6. Časy uváděné ve čtvrtém a pátém sloupci jsou vztaženy k času výpadku linky T_0.

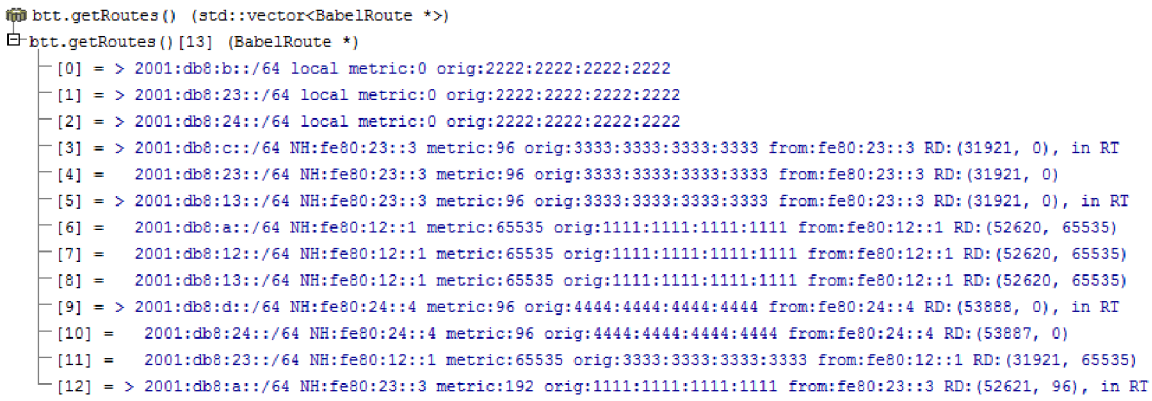
Pořadí	TLV nesené ve zprávě	Směr přenosu	Čas v simulaci [s]	Čas v realitě [s]
1	Seqno Request	R2→R3	0,187	0,208
2	Seqno Request	R3→R1 (unicast)	0,347	1,079
3	Update	R1→R3	0,595	1,152
4	Update	R3→R2	0,673	1,275

Tabulka 6: Komunikace mezi směrovači R1, R2 a R3 krátce po výpadku

První zprávou směrovač R2 požaduje navýšení sekvenčního čísla pro daný prefix. Tuto zprávu odesílá vícesměrově na všech rozhraních. Při jejím přijetí R3, je ověřeno, zda může takovýto požadavek obsloužit. Jelikož však není původcem požadovaného prefixu, pouze jej přepošle dál, a to adresovaného přímo sousednímu směrovači, od kterého získal informaci o daném prefixu. Tomu v tabulce Tabulka 6 odpovídá zpráva č. 2. R1 reaguje na přijetí požadavku shodně, ovšem na rozdíl od R3 je původcem prefixu a proto zvyšuje své sekvenční číslo a odpovídá aktualizací ve zprávě 3. Po doručení této odpovědi R3 zjistí ze záznamu v tabulce nevyřízených požadavků, že se jedná o odpověď na přeposlaný požadavek. Z toho důvodu je ve čtvrté zprávě tato informace s adekvátní úpravou metriky přeposlána původnímu odesílateli.

Jak je dokumentováno obrázkem Obrázek 6.8 a tabulkou Tabulka 7 výsledná topologie v simulačním prostředí je shodná s programem *babeld*. Ovšem rozdíl je možné pozorovat v čase zasílání

druhé zprávy. Tento rozdíl přikládám buď mechanismu dočasného pozdržení zpráv, nebo zvýšenému provozu na síti krátce po výpadku.



Obrázek 6.8: Tabulka topologie směrovače R2 v simulačním prostředí, v době po získání informací o jiných cestách do cíle

Prefix	Metrika	Ohlášená metrika	RouterID původce	Sekvenční číslo	Odchozí rozhraní	Následující skok	Příznak
2001:db8:b::/64	0						(exported)
2001:db8:12::/64	0						
2001:db8:23::/64	0						(exported)
2001:db8:24::/64	0						(exported)
2001:db8:a::/64	192	96	0a:00:00:ff:fe:00:12:01	36153	eth3	fe80::a00:ff:fe00:2303	(installed)
2001:db8:a::/64	65535	0	0a:00:00:ff:fe:00:12:01	36152	eth0	fe80::a00:ff:fe00:1201	
2001:db8:c::/64	96	0	0a:00:00:ff:fe:00:13:03	46991	eth3	fe80::a00:ff:fe00:2303	(installed)
2001:db8:d::/64	96	0	0a:00:00:ff:fe:00:24:04	4936	eth2	fe80::a00:ff:fe00:2404	(installed)
2001:db8:12::/64	65535	0	0a:00:00:ff:fe:00:12:01	36152	eth0	fe80::a00:ff:fe00:1201	(feasible)
2001:db8:13::/64	96	0	0a:00:00:ff:fe:00:13:03	46991	eth3	fe80::a00:ff:fe00:2303	(installed)
2001:db8:13::/64	65535	0	0a:00:00:ff:fe:00:12:01	36152	eth0	fe80::a00:ff:fe00:1201	(feasible)
2001:db8:23::/64	96	0	0a:00:00:ff:fe:00:13:03	46991	eth3	fe80::a00:ff:fe00:2303	(feasible)
2001:db8:23::/64	65535	96	0a:00:00:ff:fe:00:13:03	46991	eth0	fe80::a00:ff:fe00:1201	(feasible)
2001:db8:24::/64	96	0	0a:00:00:ff:fe:00:24:04	4936	eth2	fe80::a00:ff:fe00:2404	(feasible)

Tabulka 7: Tabulka topologie směrovače R2 v reálném prostředí, v době po získání informací o jiných cestách do cíle

7 Závěr

Tato práce se věnuje rozšíření simulačního prostředí OMNeT++ o podporu směrovacího protokolu Babel.

V úvodu jsou předloženy důvody, které vedly k vytvoření této práce. Druhá kapitola podrobně rozebírá směrovací protokol Babel. Detailně popisuje nežádoucí situace, ke kterým může v sítích docházet, a mechanismy, umožňující tyto situace řešit, či jim předcházet. Dále uvádí koncepci a činnost protokolu, doplněnou o popis používaných datových struktur. V závěru této kapitoly jsou uvedeny existující implementace. Je popsán jejich stav, kompatibilita s operačními systémy, způsob konfigurace a podpora rozdílných strategií výpočtu ceny linky. Ve třetí kapitole je čtenář zasvěcen do problematiky simulace počítačových sítí v prostředí OMNeT++. Je diskutován způsob rozšiřování tohoto prostředí o podporu nových protokolů. Čtvrtá kapitola předkládá návrh implementace. Ten zahrnuje hierarchii modelu a strukturu konfiguračního souboru. Popis implementace, je obsahem kapitoly páté. V ní jsou popsány jednotlivé moduly a datové typy jimi používané. Následuje ověření správnosti porovnáním s existující implementací, při kterém jsou srovnávány jak výsledné topologie, tak i průběh komunikace.

Jako součást práce bylo vytvořeno rozšíření simulačního modelu směrovacího protokolu EIGRP. Výsledky práce na tomto rozšíření budou publikovány ve článku *Enhanced Interior Gateway Routing Protocol with IPv4 and IPv6 Support for OMNeT++* časopisu *Advances in Intelligent Systems and Computing* nakladatelství *Springer-Verlag*.

Dalším pokračováním na této práci může být její rozšiřování o podporu nových strategií hodnocení ceny linky nebo výběru cest.

Literatura

[1] CHROBOCZEK, Juliusz. *RFC 6126: The Babel Routing Protocol* [online]. 2011 [cit. 2015-01-14]. Dostupné z: https://tools.ietf.org/html/rfc6126

[2] CHROBOCZEK, Juliusz. *Babel: A flexible routing protocol* [online]. 2014 [cit. 2015-01-14]. Dostupné z: http://www.pps.univ-paris-diderot.fr/~jch/software/babel/babel-20140311.pdf

[3] CHROBOCZEK, Juliusz. *Internet-Draft: Extension Mechanism for the Babel Routing Protocol* [online]. 3. vyd. 2014 [cit. 2015-01-14]. Dostupné z: https://tools.ietf.org/html/draft-chroboczek-babel-extension-mechanism-03

[4] OVSIENKO, Denis. *RFC 7298: Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication* [online]. 2014 [cit. 2015-01-14]. Dostupné z: https://tools.ietf.org/html/rfc7298

[5] CHROBOCZEK, Juliusz. *Babel: A routing protocol for sparse networks* [online]. 2008 [cit. 2015-01-14]. Dostupné z: http://www.pps.univ-paris-diderot.fr/~jch/software/babel/babel-funkfeuer.pdf

[6] MURRAY, David, Michael DIXON a Terry KOZINIEC. *An Experimental Comparison of Routing Protocols in Multi Hop Ad Hoc Networks* [online]. Australasian Telecommunication Networks and Applications Conference, 2010 [cit. 2015-01-14]. Dostupné z: http://researchrepository.murdoch.edu.au/3982/1/Comparison_of_Routing_Protocols.pdf

[7] ABOLHASAN, M., B. HAGELSTEIN a J. C.-P. WANG. *Real-world Performance of Current Proactive Multi-hop Mesh Protocols* [online]. Shanghai, China: IEEE Asia-Pacific Conference on Communication, 2009 [cit. 2015-01-14]. Dostupné z: http://ro.uow.edu.au/cgi/viewcontent.cgi?article=1747&context=infopapers

[8] TANG, Linpeng a Qin LIU. *A Survey on Distance Vector Routing Protocols* [online]. 2011 [cit. 2015-01-14]. Dostupné z: http://arxiv.org/abs/1111.1514v1

[9] HAUCK, Antoine a Peter SOLLBERGER. *Babel Multi-hop Routing for TinyOS Low-power Devices* [online]. [cit. 2015-01-14]. Dostupné z: http://blog.antoine.li/files/2011/09/babel_paper.pdf

[10] GARCIA-LUNES-ACEVES, J. Loop-Free Routing Using Diffusing Computations. *IEEE/ACM Transactions on Networking*. 1993, roč. 1, č. 1, s. 130-141.

[11] BERTSEKAS, Dimitri a Robert GALLAGER. *Data Networks* [online]. 2. vyd. Prentice Hall, 1992 [cit. 2015-01-14]. ISBN 0132009161. Dostupné z: http://web.mit.edu/dimitrib/www/datanets.html

[12] ČERNÝ, Jakub. *Základní grafové algoritmy* [online]. 2010 [cit. 2015-01-14]. Dostupné z: http://kam.mff.cuni.cz/~kuba/ka/ka.pdf

[13] BLOUDÍČEK, Jan. Modelování směrovacího protokolu EIGRP. Brno, 2014. Diplomová práce. FIT VUT v Brně

[14] VESELÝ, Vladimír, Jan BLOUDÍČEK a Ondřej RYŠAVÝ. *Enhanced Interior Gateway Routing Protocol for OMNeT++*. Proceedings of the 4th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2014). SciTePress, Wien, 2014 [cit. 2015-01-14]. Dostupné z: http://www.fit.vutbr.cz/units/UIFS/pubs/all.php?file=%2Fpub%2F10739%2Fclanek-cr2.pdf&id=10739

[15] *Git repozitář: kvetak/ANSA* [online]. [cit. 2015-01-14]. Dostupné z: https://github.com/kvetak/ANSA

[16] *INET Framework for OMNeT++: Manual* [online]. 2012 [cit. 2015-01-14]. Dostupné z: http://inet.omnetpp.org/doc/INET/inet-manual-draft.pdf

[17] *OMNeT++ User manual 4.6* [online]. [cit. 2015-01-14]. Dostupné z: http://www.omnetpp.org/doc/omnetpp/manual/usman.html

[18] *Git repozitář: Quagga-RE/quagga-RE* [online]. [cit. 2015-01-14]. Dostupné z: https://github.com/Quagga-RE/quagga-RE

[19] CHROBOCZEK, Juliusz. *Git repozitář: jech/babeld* [online]. [cit. 2015-01-14]. Dostupné z: https://github.com/jech/babeld

[20] ISO/IEC 7498-1. *Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. 2. vyd. 1994.

[21] CHROBOCZEK, Juliusz. The babeld(8) manual page. [online]. [cit. 2015-01-14]. Dostupné z: http://www.pps.univ-paris-diderot.fr/~jch/software/babel/babeld.html

[22] PERKINS, Charles a Pravin BHAGWAT. *Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers*. ACM SIGCOMM Computer Communication Review. 1994, roč. 24, č. 4, s. 234-244, [cit. 2015-01-14]. Dostupné z: http://www.cs.virginia.edu/~cl7v/cs851-papers/dsdv-sigcomm94.pdf

[23] PERKINS, Charles, Elizabeth BELDING-ROYER a Samir DAS. *RFC 3561: Ad hoc On-Demand Distance Vector (AODV) Routing* [online]. 2003 [cit. 2015-01-14]. Dostupné z: https://tools.ietf.org/html/rfc3561

[24] *Enhanced Interior Gateway Routing Protocol* [online]. 2005, aktualizováno 2015 [cit. 2015-01-14]. Document ID: 16406. Dostupné z: http://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.pdf

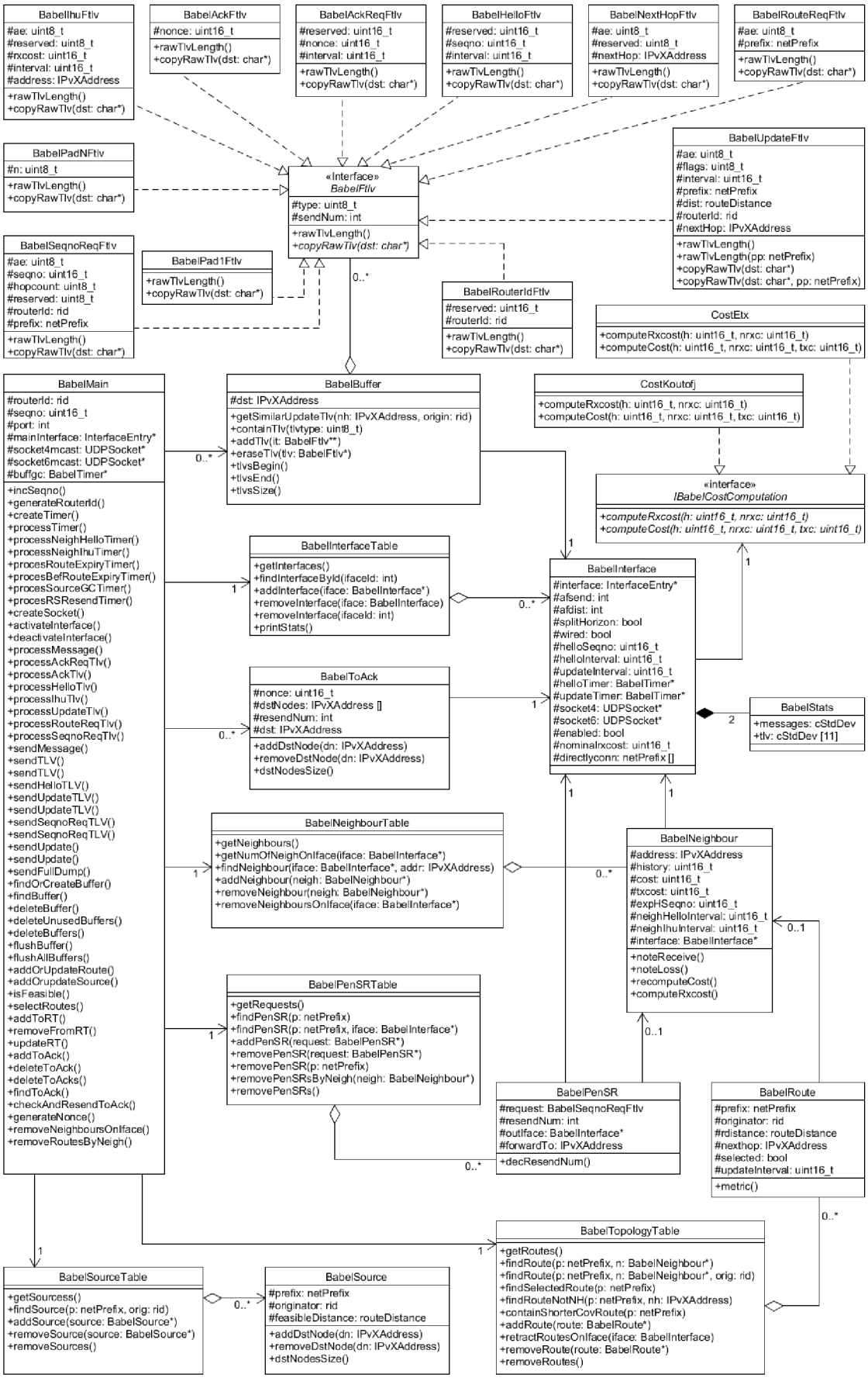
[25] DE COUTO, Douglas S. J. *High-Throughput Routing for Multi-Hop Wireless Networks* [online]. 2004 [cit. 2015-01-14]. Dostupné z: http://pdos.csail.mit.edu/papers/grid:decouto-phd/thesis.pdf

[26] CHROBOCZEK, Juliusz. *Internet-Draft: Diversity Routing for the Babel Routing Protocol* [online]. 0. vyd. 2014 [cit. 2015-01-14]. Dostupné z: https://tools.ietf.org/html/draft-chroboczek-babel-diversity-routing-00

[27] JONGLEZ, Baptiste a Juliusz CHROBOCZEK. *Internet-Draft: Delay-based Metric Extension for the Babel Routing Protocol* [online]. 2014 [cit. 2015-01-14]. Dostupné z: https://tools.ietf.org/html/draft-jonglez-babel-rtt-extension-00

[28] BOUTIER, Matthieu a Juliusz CHROBOCZEK. *Internet-Draft: Source-Specific Routing in Babel* [online]. 2014 [cit. 2015-01-14]. Dostupné z: https://tools.ietf.org/html/draft-boutier-babel-source-specific-00

Diagram tříd



Obsah CD

Adresář	
/examples	Demonstrační úlohy pro ukázku funkcionality výsledného modelu. Zahrnují předpřipravené konfigurace pro různé přenosové technologie.
/src-babel	Zdrojové soubory výsledného simulačního modelu protokolu Babel
/src-ansainet	Zdrojové kódy knihovny ANSAINET včetně podpory protokolu Babel
/text	Text práce a diagram tříd v editovatelném formátu
/verify-tests	Data získaná při provádění validačních testů v kapitole 6. Zahrnují výstupy simulací, logy programu babeld a záznamy síťové komunikace.

Konfigurace programu babeld

Klíčové slovo konfiguračního souboru	Parametr programu	Vysvětlivka
protocol-group *group*	-m	Multicastová adresa skupiny, používaná pro zasílání zpráv
protocol-port *port*	-p	UDP port, používaný při zasílání zpráv
kernel-priority *priority*	-k	Priorita používaná při přidávání cest do jádra
allow-duplicates *priority*		Umožnění duplikace cest s prioritou minimálně *priority*
keep-unfeasible {true|false}	-u	Určuje, zda budou nevhodné cesty udržovány v paměti
random-id {true|false}	-r	Nastavuje použití náhodného identifikátoru směrovače
debug *level*	-d	Určuje úroveň ladících výpisů
local-port *port*	-g	TCP port, na kterém program očekává připojení grafické nadstavby
export-table *table*	-t	Směrovací tabulka používaná pro ukládání cest
import-table *table*	-T	Směrovací tabulka, jejíž cesty jsou redistribuovány
link-detect {true|false}	-l	Použití CS pro určení dostupnosti rozhraní
diversity {true|false|kind}		Algoritmus pro určování cest rušených bezdrátových spojů
diversity-factor *factor*		Faktor určující zvýhodnění nerušených bezdrátových spojů
smoothing-half-life *seconds*	-M	Poločas přeměny exponenciálního zpoždění, použitý při zohledňování historie metrik
deamonise {true|false}	-D	Program poběží jako démon
state-file *filename*	-S	Umístění souboru obsahujícího stav
log-file *filename*	-L	Umístění souboru obsahujícího log
pid-file *filename*	-I	Umístění souboru obsahujícího identifikátor procesu
interface *name* [*parameter*]		Konfigurace síťového rozhraní name, možné parametry jsou uvedeny v tabulce Tabulka 9
default *name* [*parameter*]		Výchozí konfigurace síťových rozhraní, možné parametry jsou uvedeny v tabulce Tabulka 9

Tabulka 8: Klíčová slova konfiguračního souboru programu *babeld*

Klíčové slovo konfiguračního souboru	Vysvětlivka
wired {true|false|auto}	Optimalizace specifická pro drátové rozhraní
link-quality {true|false|auto}	Odhad kvality linky, ve výchozím nastavení použito pouze na bezdrátových rozhraních
split-horizon {true|false|auto}	Použití mechanismu split-horizon, ve výchozím nastavení na bezdrátových rozhraních vypnuto
rxcost *cost*	Určuje cenu příjmu zpráv za ideálních podmínek, výchozí hodnota je pro drátové sítě 96, pro bezdrátové 256
channel *channel*	Kanál používaný bezdrátovým rozhraním, ve výchozím nastavení je detekován automaticky
faraway {true|false}	Určuje, zda je síť dostatečně daleko, aby nedocházelo k rušení, ve výchozím nastavení vypnuto
hello-interval *interval*	Interval zasílání Hello zpráv
update-interval *interval*	Interval zasílání periodických aktualizací cest
enable-timestamps {true|false}	Odesílání časových razítek se všemi zprávami Hello a IHU, pro výpočet RTT
rtt-decay *decay*	Faktor úpadku, hodnota mezi 1 a 256, včetně
rtt-min *rtt*	Minimální RTT v milisekundách, od které je cena linky penalizována výchozí 10 ms
rtt-max *rtt*	Maximální RTT v milisekundách, po kterou je cena linky penalizována, výchozí 120 ms
max-rtt-penalty *cost*	Maximální penalizace ceny linky z důvodu zpoždění linky, ve výchozím nastavení 0

Tabulka 9: Klíčová slova parametrů konfigurace síťových rozhraní pro program *babeld*

Seznam zkratek

ANSA - Automated Network Simulation and Analysis

AODV - Ad hoc On-Demand Distance Vector

ARP - Address Resolution Protocol

BSD - Berkeley Software Distribution

DEVS - Discrete Event System Specification

DSDV - Destination-Sequenced Distance-Vector

EIGRP - Enhanced Interior Gateway Routing Protocol

ETX - Estimated Transmission Cost

EUI-64 - Extended Unique Identifier - 64

FC - Feasibility Condition

FD - Feasibility Distance

FEL - Future Event List

GC - Garbage Collection

IANA - Internet Assigned Numbers Authority

IDE - Integrated Development Environment

IHU - I Heard You

IPv4 - Internet Protocol version 4

IPv6 - Internet Protocol version 6

IS-IS - Intermediate System to Intermediate System

ISO/OSI - International Organization for Standardization/Open Systems Interconnection

IT - Interface Table

MIT - Massachusetts Institute of Technology

MPI - Message Passing Interface

ND - Neighbor Discovery

NED - Network Description

NT - Neighbour Table

OSPF - Open Shortest Path First

PDM - Protocol Dependent Module

PIM-SM - Protocol Independent Multicast - Sparse Mode

PPP - Point-to-Point Protocol

RD - Reported Distance

RFC - Request for Comments

RID - Router Identifier

RIP - Routing Information Protocol

RIPng - Routing Information Protocol next generation

RT - Routing Table

RTT - Round-Trip Time

SCTP - Stream Control Transmission Protocol

SNC - Source Node Condition

ST - Source Table

TCP - Transmission Control Protocol

TLV - Type-Length-Value

TRILL - Transparent Interconnection of Lots of Links

TT - Topology Table

UDP - User Datagram Protocol

VRRP - Virtual Router Redundancy Protocol

XML - Extensible Markup Language