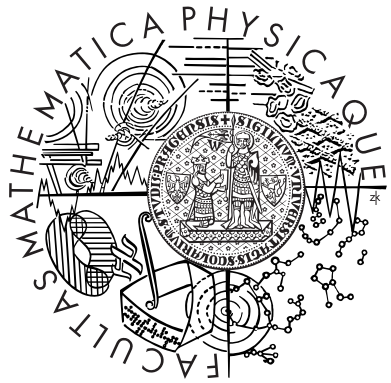
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Martin Káldy

Algoritmy pro směrování v mobilních sítích

Katedra aplikované matematiky

Vedoucí bakalářské práce: Prof. RNDr. Luděk Kučera, DrSc.

Studijní program: Informatika, programování

2006

Poděkování

Děkuji svému vedoucímu práce, Prof. RNDr. Luďkovi Kučerovi DrSc. za cenné rady k psaní této práce a správné nasměrování už při vytváření souvisejícího ročníkového projektu.

Prohlašuji, že jsem svou bakalářskou práci napsal sám a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze, dne Martin Káldy

Obsah

1 Úvod **6**

1.1 Mobilní ad hoc sítě 6

1.2 Model vrstev 7

1.3 Routovací algoritmy 7

1.4 Terminologie 9

2 Ad-hoc On-demand Distance Vector **10**

2.1 Cykly, signatury a sekvenční čísla 11

2.2 Routovací tabulky 12

2.3 Druhy paketů AODV 13

2.4 Vyhledávání cesty 14

2.5 Výpadky spojení a stárnutí cest 15

2.6 Jednosměrné cesty 17

2.7 Lokální konektivita 17

2.8 Analýza AODV 17

3 Modul pravděpodobnostní redukce floodingu **18**

3.1 Problém floodingu a idea optimalizace 18

3.2 Postup redukce 19

3.3 Podhodnocená vlna a resend fáze 20

3.4 Poznámky k využitelnosti modulu 20

4 Modul mostů **20**

4.1 Problém výpadků cest 20

4.2 Rozšířená signatura 21

4.3 Nové typy paketů 22

4.4 Vytvoření mostu 22

4.5 Konsolidace cesty 23

4.6 Kombinace modulů 24

4.7 Poznámky k využitelnosti modulu 25

5 Modul route optimalizace **25**

5.1 Idea optimalizace 25

5.2 Přeposílání optimalizačního paketu 26

6 Simulace **26**

6.1 MOBNET 26

6.2 Simulační model 27

6.3 Poznámky k implementaci routovacího algoritmu 28

6.4 Výsledky simulace 29

7 Závěr **30**

8 Příloha: Grafy **32**

Jméno práce: Algoritmy pro směrování v mobilních sítích

Autor: Martin Káldy

Katedra (ústav): Katedra aplikované matematiky

Vedoucí bakalářské práce: Prof. RNDr. Luděk Kučera, DrSc.

E-mail vedoucího: ludek@kam.mff.cuni.cz

Abstrakt: Cílem práce je navrhnout routovací algoritmus pro mobilní ad hoc sítě, který by byl schopen fungovat v rozsáhlých sítích s velkou hustotou uzlů a naopak s poměrně nízkým datovým provozem. Algoritmus by se měl být schopen vypořádat i s vyšší mobilitou uzlů a s častými výpadky. Jako základ posloužil existující protokol AODV. Definujeme obecný pojem "signatury informace", abychom mohli algoritmus rozšiřovat a mohli prokázat nevytváření cyklů. Byly navrhnuty tři rozšiřující moduly a otestovány v simulačním prostředí MOBNET. Výsledný algoritmus prokázal ve ztížených přenosových podmínkách výrazně lepší výsledky.

Klíčová slova: Mobilní ad hoc sítě, Směrování paketů, Ad hoc on demand distance vector, Signatura informace, Simulace provozu sítě.

Title: Algorithms for Routing in Mobile Networks

Author: Martin Káldy

Department: Department of Applied Mathematics

Supervisor: Prof. RNDr. Luděk Kučera, DrSc.

Supervisor's e-mail address: ludek@kam.mff.cuni.cz

Abstract: This thesis is focused on design of a routing algorithm, that would be able to operate in large networks with high density of nodes and with relatively low data traffic. The algoritm have to be effective with high mobility rate of nodes and with frequent link breakages. The algorithm is based on existing routing protocol AODV. We define the general concept of "information signature" in order to improve the algorithm and to prove loop freedom. We present three modules and test them in the MOBNET simulation environment. The improved algorithm has proven significantly higher performance under difficult routing conditions.

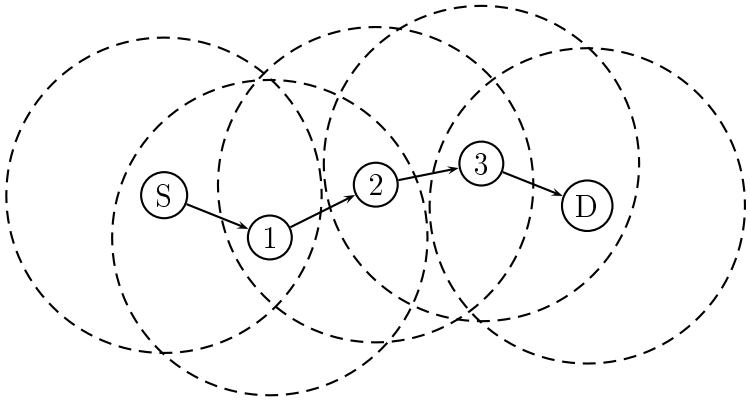
Keywords: Mobile ad hoc networks, Routing, Ad hoc on demand distance vector, Information signature, Network simulation

1 Úvod

V první kapitole vymezíme pojem mobilní ad hoc sítě a prozkoumáme různé přístupy k mobilnímu routování. V druhé kapitole se zaměříme na základní protokol AODV. Zavedeme pojem signatury informace a popíšeme přesně algoritmus, z kterého budeme později vycházet. Tento algoritmus podrobíme analýze a pokusíme se určit potenciální slabiny. V třetí, čtvrté a páté kapitole definujeme tři rozšiřující moduly. Jedná se o modul flood redukce, modul mostů a modul route optimalizace. V šesté kapitole uvedeme simulační projekt MOBNET a vyhodnotíme simulace. Grafy jsou uvedeny v příloze.

1.1 Mobilní ad hoc sítě

MANET je zkrácené označení pro tzv. mobilní ad-hoc sítě (mobile ad-hoc networks). Jsou to sítě typu peer-to-peer kde uzly tvoří mobilní stanice, které mezi sebou komunikují bezdrátově. Každá stanice se chová jako host i jako router- zachycuje a přeposílá pakety aby umožnila komunikovat dalším stanicím, jež jsou od sebe vzdáleny dále než je dosah jejich signálu. V obecném případě, pakety vyslané stanicí musí urazit několik skoků (hops) než dorazí ke svému cíli. Na obrázku vyznačeno přeposílání paketu a dosah signálu stanic. Na rozdíl od klasických sítí, v ad-hoc sítích neexistují pevné spoje s dlouhodobou spolehlivostí. Komunikovat přímo mohou libovolné stanice, které jsou dost blízko, aby se nacházely v dosahu svého signálu. Je zřejmé, že ad-hoc systémy se musí nějakým způsobem samy v reálném čase konfigurovat a musí být dostatečně přizpůsobivé k výpadkům, ke změnám topologie spojení a k připojování a odpojování stanic.



Obrázek 1: Multi-hop přenos. Kruhy značí dosah signálu

Současné sítě mobilních telefonů nejsou MANET. Mobilní stanice se připojují k pevným routerům (bázovým stanicím), které svým signálem pokrývají určitou oblast (buňku) a mezi sebou jsou již dále spojeny jako síť se statickou strukturou. Mobilní stanice mohou komunikovat tehdy, nachází-li se obě v oblastech pokrytých signály bázových stanic. Je třeba podotknout, že na pevnou síť (především internet) mo-

hou být napojeny i ad-hoc soustavy. Připojení pak opět probíhá obecně přes několik skoků.

1.2 Model vrstev

Síťová komunikace se zpravidla rozděluje do sedmi (model ISO OSI [13]) základních vrstev. Každá vrstva má vymezenou úlohu a při plnění se opírá o funkce nižší vrstvy. Stručně popsáno, Fyzická vrstva se stará o fyzikální reprezentaci a přenos dat bez ohledu na jejich obsah. Linková vrstva řeší přímé spojení dvou sousedních uzlů a přístup ke komunikačnímu médiu. Úkolem síťové vrstvy je zařídit přenos dat mezi vzdálenými uzly. To zahrnuje hledání přenosové cesty (route) přes několik mezičlánků (sítí a routerů) a její udržování. Další, transportní vrstva, pak zajišťuje spojení mezi koncovými uzly a jeho spolehlivost. Dále existuje relační vrstva pro navazování a ukončování relace, prezentační vrstva pro kódování a šifrování dat a poslední, aplikační vrstva, která implementuje rozhraní pro síťové služby.

Tato práce se zabývá síťovou vrstvou ad-hoc sítě. Chceme, aby na požádání zprostředkovala výměnu dat mezi vzdálenými uzly, a to metodou best-effort, tedy s úsilím o co nejefektivnější, nejrychlejší, nejúspornější, ale ne nutně spolehlivý přenos. Algoritmus se nezaobírá významem a reprezentací přenášených dat a nechává na vyšších vrstvách, aby se postaraly o kvalitu spojení (například náprava ztráty paketů).

Pro funkčnost routovacího algoritmu předpokládáme následující služby linkové vrstvy: Poslání paketu konkrétnímu adresátovi a případné vygenerování chyby, pokud se nelze spojit, a dále pak broadcastování paketu všem stanicím v dosahu signálu. Rámce linkové vrstvy poskytují identifikaci odesílatele zprávy kompatibilní s identifikátory na síťové vrstvě. Počítáme s tím, že spojení bude umožněno až od určité hranice spolehlivosti, kterou převedeme na vzdálenost a budeme ji při hledání cesty brát jako dosah signálu.

1.3 Routovací algoritmy

Routování paketů se děje přes routovací tabulky. Tabulky definují, jak podle adresáta přeposílat pakety dalším uzlům. V pevných sítích není obtížné udržovat tabulky aktuální díky statické topologii spojů. Tabulky se nemusí obnovovat často a proto jsou aktualizační náklady poměrně nízké.

Oproti tomu v mobilních ad-hoc sítích se topologie mění rychle. Tabulky by v ideálním případě (kompletní data o síti) měly mít velikost $O(N)$, označíme-li N počet mobilů. Informaci o každém uzlu je třeba distribuovat mezi všechny ostatní uzly, takže jednorázově by šlo o množství přenesených kontrolních dat $\Omega(N^2)$[1], Očekáve-

[1]Nejúspornější unicastové rozesílání by proběhlo po stromovém grafu. Při přeposlání údajů o každém uzlu všem ostatním dostáváme počet $N.(N-1)$ přeposlání jednotlivých záznamů. Samozřejmě nezávisle na počtu, "po kolika" data posíláme. Na jednu stranu je výhodné posílat routovací informace pospolu ve velkých paketech, avšak vytvořené cesty pak nejsou ideální. Charakteris-

jme tudíž na každém uzlu provoz $O(N)$. Při konstantní hustotě mobilních stanic rozmístěných do určitého tvaru se budou vytvářet spojení vyžadující $O(\sqrt{N})$ skoků.

Současné routovací algoritmy a přístupy k routování v ad-hoc sítích lze rozdělit do tří kategorií [4]: jednoduché (s podskupinami proaktivní a reaktivní), hierarchické a geografické.

Proaktivní jednoduché algoritmy se snaží udržovat úplnou či dostačující aktuální informaci o síti nezávisle na skutečném provozu. Proaktivní přístup je vhodný pro menší sítě s velkým provozem a požadavkem na kvalitní spojení. Příliš velká síť bude mít však tendenci zahltit se vlastní údržbou. Příklady proaktivních algoritmů jsou Optimized Link-State Routing (OLSR) a Fisheye State Routing (FSR).

Reaktivní jednoduché routování nevytváří žádný provoz, dokud není požadavek na přenos dat. Jakmile se požadavek objeví, algoritmus musí najít a sestavit cestu.Protokoly mají dvě základní části: Nalezení spojení (route discovery), které se obvykle provádí přes flooding a udržování (maintenance). Reaktivní přístup vyhovuje velkým sítím s velkou mobilitou uzlů a malým provozem. Mezi příklady uveďme Dynamic Source Routing (DSR), standartní Ad-hoc On-demand Distance Vector (AODV, RFC3561) a Virtual Path Routing (VPR).

Hierarchické routování. Jednoduché routování nerozlišuje funkci uzlů. Oproti tomu hierarchické routování se snaží vytvářet hierarchii uzlů a jejich virtuální strukturu tak, aby se snížily routovací nároky. Lze zhruba hodnotit jako vhodné pro velmi rozsáhlé sítě s nízkou mobilitou uzlů. Příkladem je Hierarchical State Routing (HSR) nebo Clusterhead Gateway Switch Routing (CGSR). Hierarchické routování se zhusta používá pro nepohyblivé sítě, například internet.

Geografické algoritmy počítají s tím, že jsou mobilní stanice vybaveny přístrojem na určování polohy (GPS). Díky tomu se snaží optimalizovat hledání cesty nebo posílat data vpřed "naslepo" pouze díky znalosti polohy cílového uzlu. Jako příklad uveďme Greedy Perimeter Stateless Routing (GPSR), Location Aided Routing (LAR) a Geographical Temporally Ordered Routing Algorithm (GeoTORA).

Snažíme se navrhnout algoritmus vhodný i pro rozsáhlé sítě s nižším datovým provozem a netriviální mobilitou uzlů. Geografické routování se zabývá metodami, jak zužitkovat geografickou informaci [6, 11]. Fáze route discovery je ale buďto přeskočena (informace o poloze cílového mobilu je známa předem), nebo implementována proaktivním či reaktivním floodingem.

Kromě geografických algoritmů se zdá nejvýhodnější reaktivní jednoduché (on-demand) routování. V této kategorii máme několik algoritmů. Algoritmus Dynamic Source

tiky takové sítě se rozebírají v [8]. Dále při použití broadcastu můžeme počet rozesílání snížit až několikrát podle hustoty mobilů.

Routing [5] volí cesty přímo u zdroje a zaznamenává ji do hlavičky paketu. Pro delší cesty se paket zvětšuje a opravy cesty se musí řešit u zdroje. Virtual Path Routing [1] vyhledává cíl unicastově rozesílanými pakety s pomocí několika "virtuálních cest", dřív než přistoupí k floodingu. Pro rozšiřování je tento algoritmus dost složitý. V této práci se budeme rozšiřovat routování Ad hoc On demand Distance Vector [10]. O přeposílání rozhoduje každý uzel na cestě podle "následníka" (nexthop) uvedeného v routovací tabulce. Tento algoritmus volíme proto, že v rozšíření může právě uzel podle situace cestu měnit, opravovat apod., a tím okamžitě "na místě" reagovat na změny v síti.

1.4 Terminologie

Mobil: mobilní stanice, uzel v síti. Mobily se chovají jako uživatelé i jako routery. Jsou si vpodstatě rovnocenné, avšak v hierarchických protokolech mohou zastávat různé funkce.

Paket: balík dat na síťové úrovni. Rozlišujme datový paket, přeposílající "užitečná data" vyšších vrstev a kontrolní paket který obsahuje informace pro řízení spojení. Základní údaje paketu jsou mimo jiné typ zprávy, zdroj, cíl a time to live.

Rámec: balík dat na linkové vrstvě. Paket obalený dalšími informacemi, mimo jiné odesílatelem a příjemcem.

Source, Originator, Zdroj: uzel sítě který si vyžádal spojení.

Destination, Cíl: cílový uzel spojení. Pro packety vyslané opačným směrem se zdroj a cíl nepřehazují. Pokud uzel zachytí zprávu, zpravidla napřed zkontroluje, zda sám není cílem, a podle toho zprávu zpracuje nebo přepošle dál (forwarduje)

Odesílatel: bezprostřední odesílatel zprávy, identifikovaný na linkové vrstvě, uveden nezávisle na typu zprávy

Příjemce: bezprostřední příjemce zprávy, identifikovaný na linkové vrstvě, uveden nezávisle na typu zprávy

Time to live (TTL): počet hopů, které smí paket ještě urazit. Při každém přeposlání se snižuje o 1 a pokud je na nule, pak se místo forwardování zahazuje.

Flood, flooding: broadcast paketu přes celou síť. Aby se nepřehlcovala síť, flood pakety mají jen určité TTL a dále jsou označeny flood-ID. Jakmile uzel zachytí paket, jehož flood-ID má mezi zpracovanými, paket ihned zahazuje. Flood-ID se generuje spojením ID uzlu a unikátním číslem (stačí unikátnost v dostatečném časovém měřítku) v rámci tohoto uzlu.

Hop: Jeden skok v síti, tedy jedno přeposlání mezi sousedy. Hopcount značí počet skoků k nějakému cíli.

Nexthop: Pro záznam v routovací tabulce pro přeposílání, nexthop značí bezprostředního následníka na cestě paketu. Při přeposílání paketu se zadá nexthop jako příjemce.

Sekvenční číslo: Číslo, které se při požadavku (kromě výjimek) napřed inkrementuje a pak předá. Sekvence se po vyčerpání všech čísel vrací na číselně nejnižší hodnotu. "Menší než" je pro sekvence ale definováno jako menší jen pokud je rozdíl menší než polovina rozsahu všech čísel, jinak opačně. Tím se simuluje nekonečná sekvence s korektně definovaným porovnáním. Sekvence navíc může nabývat hodnoty UNKNOWN, která je různá od každé hodnoty v rozsahu a vždy menší než tato hodnota.

Signatura: Struktura na kterou se mapuje interpretace informací přeposílaných po síti. Obsahuje sekvenční číslo které uvede poskytovatel a dodatečné čítače. Na signatuře musí být definován operátor porovnání. Platí, že při interpretaci se signatura musí zmenšit, naopak při updatování musíme použít informaci s větší signaturou. Signaturu nazvěme UNKNOWN pokud má sekvenční hodnotou UNKNOWN. Taková signatura je různá od všech ostatních signatur a je nejmenší možná.

Aktivní cesta (active route): cesta uvedená v routovací tabulce, která není označena jako "invalid" (invalidita se nastavuje po reportu o přerušení) a čas jejího vypršení ještě nenastal. Po aktivní cestě můžeme kdykoli začít vysílat pakety.

2 Ad-hoc On-demand Distance Vector

Základem práce je algoritmus AODV [10] z rodiny distance-vector směrovacích algoritmů. Routovací tabulky distance-vector udržují informaci typu

CílovýUzel - Vzdálenost (hopcount) - Nexthop

Distance-vector algoritmy jsou distribuované modifikace Bellman-Fordova algoritmu na hledání nejkratších cest v grafu. Narozdíl od předpokladů původního problému uvažujeme síť, kde se váhy spojů mohou měnit. V našem případě jde o hodnoty nekonečno nebo 1, podle toho, jak spoje vznikají a zanikají. Informace o cestě, tedy hopcount a nexthop, se šíří k sousedům. Pokud je nová cesta kratší, přepíše se (nexthop bude zdroj informace). Takto bychom v konečném čase dospěli k nejkratším cestám od každého uzlu ke každému, avšak zapříčinili bychom v síti extrémní provoz. Proto redukujeme činnost pouze na vyžádané cesty, tedy on-demand charakter.

On demand distance vector ale přenáší jen vyžádané routovací informace. Uzel hledající přenosovou cestu napřed rozešle vyhledávací vlnu paketů. Přeposílající uzly si nastavují cestu k source, nexthop volí podle toho, odkud paket přišel, tak aby znali nejkratší cestu. Pokud vlna dosáhne k destination mobilu nebo k mobilu který

zná "dostatečně aktuální" cestu k destination mobilu, pošle unicastově zpět paket odpovědi. Přeposíláním paketu odpovědi se naopak vytváří cesty k destination. Po aktivní cestě může mobil kdykoli začít vysílat pakety. Pokud se cesta přeruší, vysílá detekující uzel hlášení o chybě a cesty zneplatňuje. Aktivní cesta po stanovené době vyprší.

Poznámka: zde popisovaný algoritmus AODV byl oproti oficiální verzi pozměněn. Umožňuje tak rozšíření a přímočařejší dokazování korektnosti. Základní princip zůstává stejný, avšak liší se v detailech:

- Rozlišuje expired a invalid záznam v routovací tabulce
- Striktně požaduje vyšší signatury (viz 2.1) pro jakýkoli update
- K udržování cesty používá průběžně posílané RACK pakety

Formální důkaz dobrých vlastností (stabilita apod.) přenechejme automatickému dokazování, jak je u těchto algoritmů zvykem [2]. Provádíme pouze důkaz nevytváření cyklů při použití signatur.

2.1 Cykly, signatury a sekvenční čísla

Potenciálním problémem distance-vectoru je možnost vytvoření cyklů v síti. Existence cyklů vede k hromadění nedoručených paketů a bezúčelnému zatěžování sítě. Pokud o kolující pakety není postaráno, přeposílaly by se donekonečna a libovolně malý počet cyklů by nevyhnutelně po určité době vedl ke kolapsu sítě. Máme dvě metody, jak problém cyklů řešit. První je omezit životnost paketů, vybavit je hodnotou TTL. Zajistíme, aby kolování po určitém čase vždy skončilo. Druhá metoda je úplně zamezit, aby cykly vznikaly. Algoritmus AODV vznikání cyklů úplně zamezuje.

Cykly vznikají v zásadě proto, že uzel upravuje routovací tabulku informacemi od souseda, avšak neví, zda soused neodvodil informaci od tohoto uzlu. Řešíme problém tím způsobem, že každá routovací informace je navíc označena signaturou[2]. Pokud uzel informaci přeposílá, signatura informace se vždy *sníží*. Update povolíme tehdy, pokud máme informaci se signaturou *vyšší*. Na hodnoty signatury se mapuje závislost informací. Libovolný stupeň interpretace informace musí mít *nižší* signaturu než interpretovaná informace. AODV definuje signaturu jako sekvenční číslo a hopcount. Sekvenční číslo přiděluje při dotazu na svou polohu výhradně cílový uzel a při přeposílání se pak dále nemění. Hopcount je počet hopů, které informace urazila, inkrementuje se od nuly o 1 za každé přeposlání.

Signatura je větší, pokud:

[2]Pojem "signatury" je zaveden v rámci této práce pro účely rozšiřování algoritmu AODV. Například modul mostů tuto strukturu rozšiřuje o tzv. bridgecount, avšak základní princip zůstává stále totožný. V definici algoritmu AODV se pojem signatury neobjevuje a hovoří se odděleně o hopcountu a sekvenčním čísle.

- má větší sekvenční číslo
- má stejné sekvenční číslo a menší hopcount

Pokud uzel přepošle informaci, musí nenávratně snížit její signaturu. Nexthop upravujeme vždy na ten uzel, od kterého informace přišla. Záznam routovací tabulky lze přeposílat, pouze pokud jeho životnost (viz 2.2) nevypršela. Pokud libovolný uzel zruší záznam o aktivní cestě, má povinnost ho v routovací tabulce ještě udržovat dost dlouho, aby během této doby všem odvozeným aktivním cestám vypršela životnost a přestaly být aktivní.

Důsledek. *nexthop u aktivní cesty vždy ukazuje na uzel s vyšší signaturou routovacího údaje k danému cíli.*

Tvrzení. *Algoritmus AODV se signaturami nevytváří cyklické aktivní cesty.*

Důkaz. Pokud by se někde měl uzavřít cyklus aktivní cesty, signatura koncového bodu nové hrany by měla být vyšší než signatura počátečního. Zároveň však musí existovat aktivní cesta "po zbytku kruhu" od koncového bodu nové hrany k počátečnímu. Na této cestě se signatura musí podle důsledku stále zvyšovat. Operátor porovnání je pro signatury tranzitivní a antisymetrický, tím pádem jsme ale dospěli ke sporu.

Signatura je dostatečně obecná struktura, aby usnadnila další rozšíření. Je třeba definovat operátor (ne nutně úplného) uspořádání "menší než" a potřebujeme umět pro každou signaturu vytvořit větší i menší signaturu. Z hlediska implementace stačí, aby vlastnosti byly simulovány pro reálné podmínky (nevadí například přetékání sekvenčních čísel a omezení velikosti typů pro hopcount). Jednoduchým příkladem signatury je například reálné číslo, které na začátku zvolí uzel vytvářející informaci, a které se snižuje o 1 až 3 za každé přeposlání, podle vytíženosti přeposílajícího mobilu. Jiný příklad je rozšířená signatura popsaná v modulu mostů v sekci 4.

2.2 Routovací tabulky

AODV má charakter on-demand, takže v routovacích tabulkách shromažďuje pouze zlomek záznamů a obnovuje je pouze pokud to vyžaduje situace. Údaje v routovací tabulce mají vždy omezenou životnost. Životnost se prodlouží, pokud je údaj updatován. Pokud životnost záznamu vyprší, zůstane ještě nějakou dobu v tabulce. Doba musí být delší, než je maximální životnost updatovaného údaje. Tím zajišťujeme, aby nový údaj měl vyšší signaturu (viz 2.1). Kromě informace o životnosti má záznam flag "invalid", který je nastaven na true, pokud dorazí informace o neplatnosti spojení. Narozdíl od pouhého vypršení, invalid signalizuje neopravitelnost (viz 4). Cesta bez označení invalid a s neprošlou životností se nazývá aktivní cesta.

Záznam v routovací tabulce obsahuje následující položky:

- ID cíle cesty – v reálných sítích by byla IP adresa

- Nexthop – bezprostřední následník na cestě
- Signatura – hopcount a sekvenční číslo cíle
- Čas expirace
- Invalidita
- Seznam předchůdců (precursor list) – pouze pro sousední uzly. Seznam všech uzlů odkud vedou cesty využívající tento jako nexthop.

Tabulku updatujeme pro source i destination téměř vždy, když zpracováváme kontrolní paket. Update napřed zkontroluje, zda má nový záznam vyšší signaturu (jinak proces updatu ukončí), poté učiní záznam validní, nastaví expiraci na nový čas a zkopíruje signaturu. Update pro neexistující záznam znamená vložení nového záznamu.

2.3 Druhy paketů AODV

AODV používá 4 typy paketů: datové pakety, route request pakety (RREQ), route reply pakety (RREP) a route error pakety (RERR). Kromě RERR, všechny pakety mají uvedené source a destination.

RREQ se šíří záplavově (flood) od zdroje směrem k (neznámému) cíli. Paket obsahuje navíc data

- flood-ID (RREQ ID) – které společně se source tvoří flood-ID
- source signature – signatura cesty zpět do source
- dest signature – požadovaná minimální signatura hledané cesty
- flags – bity které blíže definují zacházení s paketem

RREP je odpovědí na RREQ, generuje jí destination nebo mobil který má aktivní cestu k destination se sekvenčním číslem dost velkým a posílá se zpět k source. Obsahuje navíc údaj

- dest signature - signatura cesty k destination

RERR je paket, který šíří informaci o výpadku spojení a nastavuje cestám flag invalid. Šíří se směrem k source a dále může být rozeslán všem předchůdcům odesílatele. Obsahuje navíc informace:

- seznam nedosažitelných destination
- sekvenční čísla nedosažitelných destination

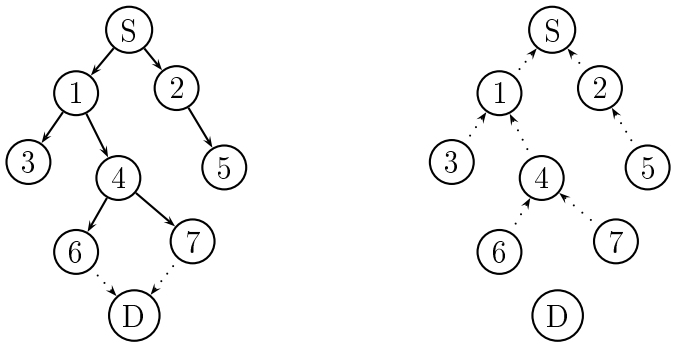
2.4 Vyhledávání cesty

Pokud v síti nikdo nechce posílat data, AODV nevykazuje žádný provoz. V případě, že chceme uskutečnit přenos a máme aktivní cestu v routovací tabulce, můžeme se pokusit rovnou přeposílat datové pakety. Konečně v případě, kdy chceme uskutečnit datový přenos, avšak nemáme aktivní cestu, musíme nějakou cestu najít a zahájíme fázi "route discovery".

Při hledání cesty se vyšle flood paketů RREQ. Source signatura dostane nové sekvenční číslo a nulový počet hopů a destination signatura bude UNKNOWN pokud neexistuje (ani neaktivní) cesta, jinak se přepíše údaj v tabulce. Signatura hledané informace musí být větší, než tento údaj. Pro flood-ID vygenerujeme nové číslo a TTL nastavíme na definovanou počáteční hodnotu. Pokus najít cestu uspěje právě tehdy, když obdržíme zpět do stanovené doby příslušný paket RREP. Pokud se tak nestane, můžeme pokus opakovat znovu s vyšším TTL. Platí pravidlo, že časový odstup mezi pokusy musí narůstat exponenciálně, abychom síť nepřetěžovali. Počet pokusů je omezený, jakmile selže poslední, algoritmus skončí oznámením chyby. Počet odeslaných vln paketů RREQ za sekundu nesmí být moc vysoký[3].

Vlna RREQ se šíří sítí a mobily si podle ní nastavují cestu zpět k source. Postupně se snižuje TTL a zvyšuje hopcount signatury k source a snižuje hopcount signatury destination. Pokud není TTL na nule, paket se broadcastově přeposílá dál. Přeposílání uspěje a zastaví se na mobilu za dvou podmínek

- Mobil je destination
- Mobil má aktivní cestu k destination se signaturou dost velkou



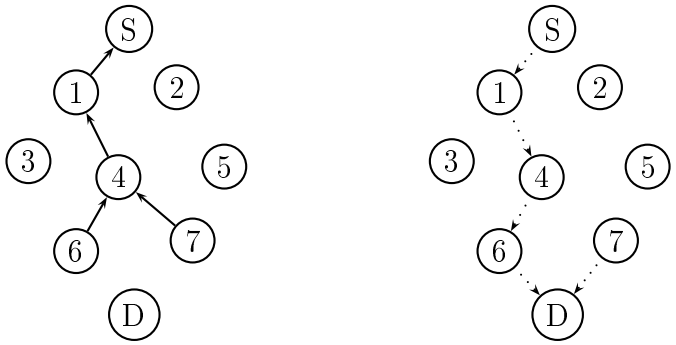
Obrázek 2: Rozesílání RREQ paketů a aktivní cesty (tečkovaně).

Obrázek 2 vlevo znázorňuje šíření RREQ paketu. Tečkovanými čarami jsou znázorněny existující aktivní cesty k destination a source a plnými přeposlané RREQ pakety. S je source, D destination. Na obrázku vpravo vidíme, jak se po rozeslání RREQ upraví aktivní cesty směrem k source.

[3]Počet mohou z bezpečnostních důvodů hlídat okolní mobily [12].

Dost velká signatura je taková, aby při přeposílání zpět měla i po přičtení všech hopů vyšší hodnotu než údaj, jenž chceme aktualizovat. Po praktické stránce toho dosáhneme tak, že KNOWN dest signatuře (na začátku je rovno údaji v routovací tabulce source mobilu) snižujeme 1 hop za každé přeposlání. S routovacím údajem uspějeme právě tehdy, když je jeho signatura větší než hodnota na obdrženém RREQ paketu. Pro UNKNOWN signaturu stačí, aby cíl měl KNOWN signaturu cesty k destination.

Při úspěšném nalezení cesty se vygeneruje paket RREP a posílá se zpět k source po cestách vytvořených RREQ floodingem. Paket RREP nese informaci o nalezené cestě a příslušnou signaturu. Signatura se zvětšuje o 1 hop za každé přeposlání. Podle přijatého RREP paketu se updatuje cesta k destination (pokud je signatura dost velká). RREP se forwarduje zpět k source v případě, že má aktivní cestu k source a proběhl update. Každým updatem mimo jiné zajišťujeme, abychom již nepřeposílali RREP pakety s nízkými signaturami. Jakmile RREP dorazí k source, může začít přenos datových paketů a dál se RREP samozřejmě nepřeposílá.



Obrázek 3: Posílání paketů RREP zpět k source a vytvořené aktivní cesty (tečkovaně)

Pokud vyžadujeme obousměrné spojení, může nám vadit případ, že o RREQ paketu se dozví uzel s aktivní cestou k destination, avšak ne destination sám. Z tohoto důvodu se při generování RREP paketu uzlem uprostřed vygeneruje ještě jeden RREP paket, který má prohozené údaje source a destination a který informuje destination o zpětné cestě k source. O tomto RREP paketu se mluví jako o "gratuitous RREP" a generuje se pokud má RREQ paket nastaven speciální flag "G".

2.5 Výpadky spojení a stárnutí cest

Po aktivní cestě se přeposílají datové pakety, dokud nejsou přenesena všechna data. Pokud linková vrstva před koncem nedokáže přeposlat paket příslušnému příjemci podle routovací tabulky, nastává tzv. výpadek spojení. Algoritmus se nestará o to, aby byla znovu přeposlána konkrétní ztracená data, to ani není úkolem síťové vrstvy, avšak musí se nějakým způsobem pokusit obnovit spojení, aby se vyšší vrstvy mohly postarat o nápravu a dokončit relaci.

V případě výpadku spojení mobil (ihned či později, více v sekci 2.7) dostane od linkové vrstvy chybovou hlášku. Potřebujeme, aby se o výpadku dozvěděl source

a mohl učinit kroky k nápravě, a případně i další mobily (nepovinně), kterých se výpadek cesty týká (mají v routovací tabulce uvedenu cestu, která je po výpadku nepoužitelná). Pro tento účel jsou broadcastovány RERR pakety, které roznáší informaci o výpadku a zneplatňují záznamy v routovacích tabulkách. Chybový stav obecně nastává ve třech případech:

- zaznamenali jsme výpadek spojení
- obdrželi jsme datový paket, ale nemáme aktivní cestu k destination (například vypršela životnost cesty)
- zachytili jsme RERR paket a máme aktivní cestu k zmíněnému destination vedoucí právě přes uzel vysílající tento RERR

V chybovém stavu zneplatníme cestu k dotyčnému destination mobilu (k dotyčným mobilům), pokud takový záznam existuje. Takovému záznamu pak nastavíme flag invalid a expiraci posuneme na aktuální čas – údaj v tabulce zůstane minimálně ještě stanovený čas, dost dlouhý, aby odvozené záznamům na sousedních mobilech mezitím vypršela životnost. Existoval-li údaj na zneplatnění, broadcastujeme sami dále paket RERR.

AODV definuje možnosti adresného posílání RERR paketů (unicast nebo několik unicastů nebo broadcast s výčtem příjemců). Výběr kandidátů na přeposlání se určuje ze seznamu předchůdců (viz 2.2). Na seznam se mobily dostávají při přeposílání RREP paketu. Vytvoří se, nebo se upraví cesty k sousedům odkud paket přišel a kam se přeposlal. Oba sousedi se navzájem přepíší do svých seznamů předchůdců pro source a destination. Paket RERR je dále vybaven seznamem nedostupných destination. Seznam se vytváří různě podle situace, která chybový stav zapříčinila. Při přeposílání RERR paketu se utvoří průnik vyjmenovaných kandidátů a cest s nexthopem nastaveným na odesílatele RERR paketu. Při výpadku spojení se výčet sestává ze všech cest s nexthopem, ke kterému spojení vypadlo. V situaci paketu bez aktivní cesty se seznam naplní právě onou nedostupnou destination.

Další variantou je úsporné rozeslání RERR unicastově pouze přímo cestě zpět k source. Pro tuto variantu bychom měli zajistit, aby cesty oběma směry byly totožné, tj. přes stejné uzly v opačném pořadí. Pro případ datového paketu bez aktivní cesty v tabulce k destination nastavme první hop RERR paketu tak, aby směřoval prioritně k odesílateli tohoto paketu, namísto k nexthopu zpětné cesty k source, pokud by se tyto adresy lišily.

Source, jakmile zachytí RERR, má zneplatněnou cestu a tudíž podle definice začne nanovo hledat cestu k destination.

Korektnost a acykličnost algoritmu není závislá na doručení RERR paketu ostatním uzlům (kromě zdroje). Pokud by nastala chyba v přenosu RERR, předchozí uzel by měl přerušené spojení zaregistrovat a sám vyslat nový RERR.

2.6 Jednosměrné cesty

Podstatné kriterium routovacího protokolu v mobilních ad-hoc sítích je schopnost vypořádat se s problémem jednosměrných spojů (unidirectional links). Hovoříme o situaci, kdy jeden uzel může přímo posílat data druhému, ale ne naopak. V grafu konektivity může dokonce nastat situace, kdy neexistuje cesta po obousměrných spojích, ale existují cesty v obou směrech po vhodně orientovaných hranách.

Algoritmus AODV neumí vytvářet cesty po jednosměrných spojích. Naopak je třeba zamezit, aby se jednosměrné spoje vůbec použily. Problém jednosměrných spojů se projeví, pokud jednosměrně přijmeme RREQ paket a přepošleme a někdy posléze je nám vrácen v protisměru RREP paket. Ten však zpět po jednosměrném spoji nedokážeme přenést a jinou cestu nemáme – nezbývá tedy než paket zahodit. Zdroj později pravděpodobně znovu pošle další vlnu RREQů, a situace se bude opakovat.

Problém se vyřeší tak, že pokud neumíme zpět předat RREP paket, pak dočasně zapíšeme daného nedosažitelného příjemce na zvláštní blacklist – od uzlů v blacklistu nepřijímáme RREQ pakety. Tak alespoň zabráníme opakování. Kontrolu na blacklist provedeme u každé zprávy RREQ těsně před updatem routovací tabulky aktivních cest. Tyto blacklisty nemají nic společného s blacklisty z oblasti zabezpečení sítě proti útokům.

2.7 Lokální konektivita

Aktivní cesta má i při perfektní funkčnosti z definice omezenou životnost. Abychom šetřili zdroje, není vhodné se vracet do fáze route discovery pokaždé, kdy základní životnost vyprší. Proto uzly na aktivní cestě proaktivně udržují spojení se sousedy. Toto pravidlo se týká pouze aktivní cesty, algoritmus se tedy pořád celkově chová reaktivně.

Základní algoritmus AODV používá pravidelné vysílání paketů HELLO sousedům, avšak pouze od uzlů na aktivní cestě. Rozšířený algoritmus AODV oproti tomu definuje pakety route acknowledgement (RACK, viz modul mostů 4 a modul route optimalizace 5), pravidelně vysílané od source i destination. Při vyslání se přidělí signatura se zvednutým sekvenčním číslem a paket updatuje cestu k odesílajícímu koncovému uzlu. Pokud RACK dlouho nepřichází, cesta vyprší a v důsledku nastane chybová situace.

2.8 Analýza AODV

Podle oficiální dokumentace [10], AODV je vhodný pro mobilní ad-hoc sítě s populací řádu desítek až tisíců mobilních stanic. Má schopnost spolehlivě fungovat i při poměrně velké mobilitě uzlů a je ideální pro sítě s nízkým provozem.

Analýza algoritmu za pomoci simulace a vizualizace ukázala na několik možných slabin. Slabiny se většinou týkají floodingu, neboť ten nejvíce zatěžuje velké sítě.

Zaprvé, flooding RREQů vykazuje špatné výsledky v oblastech s hustě rozmístěnými mobily. Protože vlnu přeposílá každý jednotlivý mobil, vzniká zbytečně veliké zahlcení (congestion) a mobily se vzájemně ruší. Bylo by vhodné, aby vlnu přeposílaly jen některé mobily.

Zadruhé: nejen nalezení, ale i každý výpadek cesty je nutno řešit novým floodingem RREQ paketů. AODV má sice metodu pro "lokální" opravu přerušených cest, avšak v řádu $O(N)$, neboť délka opravné cesty je řádově stejná jako délka cesty, tedy $O(\sqrt{N})$ a počet rozeslaných paketů bude růst s čtvercem této hrany. Delší cestu původní repair algoritmus akceptuje pouze tehdy, když má k dispozici vyšší sekvenční číslo (bližší popis v [10], sekce 6.12) – takové však zpravidla získáme až přímo od destination.

Často se jedná o lokální výpadek spoje v důsledku oddálení sousedních mobilů nebo výpadek použitelnosti nějakého mobilu po cestě. Chtěli bychom algoritmus, který by dokázal takovéto lokální problémy řešit skutečně lokálně v řádu $O(1)$ paketů, nezávisle na místě, kde se výpadek udál a bez nutnosti získávat nové sekvenční číslo.

Třetí problém je ukryt v degeneraci tvaru cesty. Pokud udržujeme dlouhou relaci a máme štěstí, že se cesta nikde nepřerušila, neznamená to, že by daná cesta byla po určitém časovém odstupu stále ještě výhodná. Pokud na druhou stranu cestu často přehodnocujeme, stojí nás to pokaždé nový flooding RREQ paketů. Chtěli bychom nalézt metodu, jak efektivně snižovat hopcount pro datové pakety.

Velký nárůst řídícího provozu a chybovosti spojené s nárůstem počtu participujících uzlů při omezené šířce pásma jsou faktory, které limitují možnosti širokého využití mobilních ad hoc sítí.

3 Modul pravděpodobnostní redukce floodingu

3.1 Problém floodingu a idea optimalizace

Zabýváme se zvyšováním efektivity floodingu v oblastech s hustým výskytem mobilů.

Připomeňme si postup zpracování vlny floodingu. Pakety mají určité TTL a dále jsou označeny flood-ID. Jakmile uzel zachytí paket, jehož flood-ID má mezi zpracovanými, paket ihned zahazuje. Flood-ID se generuje spojením ID uzlu a unikátním číslem (stačí unikátnost v dostatečném časovém měřítku) v rámci tohoto uzlu. Flood-id uchováváme dočasně ve speciální tabulce. Počet paketů na 1 vlnu roste v řádu $O(N)$.Cesta má přitom délku rostoucí v řádu $O(\sqrt{N})$. Pro velké sítě je proto důležité zaměřit se právě na úspory při floodingu.

Původní algoritmus poskytoval cestu s nejmenším počtem hopů. Pokud ustoupíme od striktního vyžadování skutečně nejkratšího spojení, můžeme díky modulu ušetřit kapacitu sítě a vyvodit několik dalších pozitivních efektů. Pro stručnost označujme pakety rozesílané floodingem jako *flood-pakety*, nebo zkráceně FP. AODV má jeden typ FP, a to RREQ, rozšíření definují ještě BREQ, viz 4.

Myšlenka optimalizace je založena na tom, že není třeba, aby každý mobil FP *přeposlal*, nýbrž aby každý mobil FP *přijal* a mohl případně odpovědět. Budeme se snažit, aby byla celá *plocha* pokrytá vysíláním flood-paketů- jejich množství by se tedy spíše odvíjelo od pokryté plochy než od počtu přítomných mobilů. Všechny informace o hustotě se navíc pokusíme získat pouze z procházejících FP. Pokud mobil registruje velkou či malou hustotu FP, sám upraví svoji pravděpodobnost přeposlání jakéhokoli dalšího FP.

3.2 Postup redukce

Pravděpodobnost přeposílání budeme odvozovat od množství zachycených broadcastů jednotlivých vln FP. Zavedeme dvě konstanty- žádoucí úroveň floodingu (kolik paketů bychom "měli" slyšet v každé vlně) a minimální úroveň floodingu (kolik paketů v každé vlně minimálně požadujeme). Konstanty by měly být nastavené tak, aby při množství větším než minimální úroveň floodingu byla okolní plocha pokrytá s dost velkou pravděpodobností. Vhodnost konstant lze testujeme při simulaci.

Pravděpodobnost přeposlání vychází primárně z množství a váhy FP předchozích vln. Váha paketu je nově zavedený údaj pro každý FP (váha je floating-point typ), a je rovna převrácené hodnotě pravděpodobnosti, se kterou se paket měl přeposlat. Váha vlny je součet vah zachycených paketů dané vlny. Váha vlny slouží jako statistický odhad střední hodnoty množství zachycených paketů v případě, že bychom modul redukce nepoužili. Pro jednotlivou vlnu je to odhad nepřesný, avšak jedná se o odhad nestranný.

Pro každou vlnu se vypočítává součet vah zachycených paketů. Suma se upravuje při přijetí paketu ve fázi kontroly, zda jsme již zpracovali pakety s daným flood-id. Pokud paket odmítneme přeposlat, jeho flood-id přesto označíme v tabulce za zpracované. Pro každý FP však upravíme váhu příslušné vlny a zapíšeme do tabulky.

Flood redukce definuje, že si mobil udržuje flood-weight-registr, který se přepíše na váhu vlny, kdykoli když váha této vlny přesáhne hodnotu v registru. Hodnota v registru již není dobrý odhad střední hodnoty váhy vlny, ale v závislosti na statistickém rozdělení hodnot je to nějaký její násobek. Výhoda tohoto postupu tkví ve schopnosti okamžitě reagovat na zvýšenou hladinu zachycených FP. Hodnota registru se časem snižuje, vždy po nějakém časovém úseku se přenásobí konstantou o něco málo menší než 1.

Pravděpodobnost přeposlání FP určíme jako podíl a žádoucí úrovně floodingu a hodnoty v registru přenásobené konstantou (jako kompenzace odhadu přes maxima). Pravděpodobnost přeposlání lze dále (ve stanovených mezích) modifikovat podle dalších ukazatelů, jako je například energie mobilu nebo aktuálního zahlcení jinými pakety. Převrácenou hodnotu pravděpodobnosti zapíšeme do FP jako jeho váhu. Pro pravděpodobnost větší než 1 přepošleme vždy a váhu FP nastavíme jako 1.

Konstanty minimální a žádoucí úrovně floodingu se mohou lišit podle typu FP.

Použitý generátor čísel musí být schopen generovat pseudonáhodná desetinná čísla v rozsahu $[0; 1)$ a seed by měl být nastaven podle IP adresy mobilu, aby generovaná čísla na jednotlivých mobilech "neinterferovala".

3.3 Podhodnocená vlna a resend fáze

Pokud se mobil rozhodne FP nepřeposlat, zařadí ho krátkodobě do tabulky nepřeposlaných FP a monitoruje počet zachycených paketů této vlny. Po vypršení časového limitu zkontroluje, že počet zachycených je větší než minimální úroveň floodingu a záznam vymaže.

V případě, že počet není dostatečný, měl by se paket opožděně přeposlat, nyní však s váhou 0. Zároveň se okamžitě sníží hodnota ve flood-registru směrem k váze dané nedostačující vlny (Utvoří se vážený průměr podle dané konstanty).

Resend fáze by se musí určitě týkat minimálně jednoho často používaného typu flood-paketů (například RREQ pro AODV) a volitelně dalších typů. Pokud mobil opustí oblast s vysokou hustotou stanic, díky resend fázi aktualizuje hodnotu v registru směrem dolů – jinak by byl odkázán na pomalé snižování popsané v předchozím oddíle.

3.4 Poznámky k využitelnosti modulu

Modul pravděpodobnostní redukce floodingu zásadně snižuje zahlcení a potřebný výkon v sítích nebo jejich částech, kde je vysoká hustota mobilních stanic. Při dostatečné hustotě navíc může mít další pozitivní důsledky ve smyslu ušetření energie a snížení zahlcení na mobilech, kde je to potřeba. Všimněme si, že úspora se netýká jen samotného přeposlání FP, ale i následné možnosti, aby přes mobil vedla aktivní cesta. Modul má ten nepříjemný důsledek, že vytvořená aktivní cesta může mít větší počet hopů.

Modul má mnoho parametrů: žádoucí úroveň floodingu a minimální úroveň floodingu (pro každý typ paketu), časový odstup pro resend fázi, konstanta na trvalé snižování hodnoty flood-weight-registru, konstanta na snižování hodnoty registru při resend fázi a (procentuální) meze individuální úpravy pravděpodobnosti přeposlání.

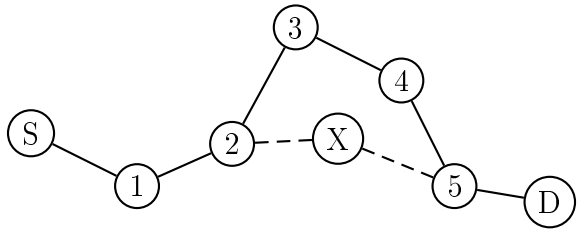
Protože čerpáme informaci o sousedních mobilech vždy z předchozích vln, chová se algoritmus stále čistě on-demand.

4 Modul mostů

4.1 Problém výpadků cest

Omezení velikosti sítě, pro kterou lze AODV efektivně použít, pramení mimo jiné ze skutečnosti, že čím delší cesta, tím spíše dojde k výpadku spojení. Pro N mobilů má

cesta mezi náhodně zvolenými mobily délku v řádu $O(\sqrt{N})$, proto i četnost výpadků lze odhadnout jako $O(\sqrt{N})$. Modul mostů sice počet výpadků nesnižuje, ale pokouší se výpadek opravit lokálně s použitím $O(1)$ paketů a v čase $O(1)$. Přitom musíme dbát na to, aby v síti nevznikaly cykly. Oprava bude mít charakter krátké "objížďky".



Obrázek 4: Vytvoření mostu přes vyřazený mobil X.

4.2 Rozšířená signatura

Abychom mohli cestu lokálně opravit, bude v mnohých případech potřeba jí prodloužit. Systém signatur z kapitoly 2.1 vylučuje, abychom cestu prodloužili, aniž bychom zvedli sekvenční číslo. Vyšší sekvenční číslo však může přidělit pouze destination a předpokládejme, že ten může být daleko. Modul mostů proto definuje jinou signaturu, původní strukturu rozšiřuje na trojici hodnot:

- sekvenční číslo - včetně možnosti UNKNOWN
- hopcount - počet klasických hopů
- bridgecount - počet hopů "po mostě"

Hopcount udává počet "standartních hopů", určený například při vytváření cesty. Nová hodnota bridgecount udává počet "mostních hopů", které vznikají pouze na lokální opravě cesty. Skutečný počet hopů je součet obou hodnot. Signatura je větší, právě tehdy když:

- má větší sekvenční číslo
- má stejné sekvenční číslo a menší hopcount
- má stejné sekvenční číslo a hopcount, ale menší bridgecount

Cesta s mostem je tedy delší, ale přípustná, protože má větší signaturu. Přitom zůstávají zachovány vlastnosti, že při přeposlání informace se signatura zmenšuje a updatujeme jen s vyšší signaturou, takže máme stále algoritmus bez cyklů.

Pokud k výpadku došlo přerušením spojení mezi dvěma mobily nebo úplným výpadkem mobilu na cestě, pravděpodobná objížďka vystačí s 3 skoky.

4.3 Nové typy paketů

Pro svou funkci definuje modul mostů tři další typy paketů. Jsou to bridge request (BREQ), bridge reply (BREP) a route acknowledgment (RACK). Pakety BREQ a BREP jsou ekvivalentní s pakety RREQ a RREP. Mají podobná data a požadují použití rozšířených signatur. Při každém hopu přičítají signaturám bridgecount namísto hopcountu. Klasické pakety nemusí přenášet celou rozšířenou signaturu, protože by měly mít nulový bridgecount, respektive ho mohou eliminovat převedením na hopcount (případ nevalidní nekonsolidované přemostěné cesty).

BREQ se šíří floodingem od mobilu, který zaznamenal výpadek spojení. Jako source je však uveden skutečný source pro danou cestu. Aby se následný BREP adresoval správnému mobilu (odkud se most staví), definuje dvě další hodnoty

- bridge-source – mobil který stavbu mostu inicioval
- bridge-source signature – signatura cesty k tomuto mobilu

BREP se šíří zpět k místu výpadku a namísto source má tedy uveden bridge-source.

RACK posílá jeden koncový mobil druhému koncovému mobilu a nese informaci o signatuře prvního zmíněného mobilu. Odpovídá RREP paketu, avšak modul route optimalizace (viz 5) může upravovat jeho vlastnosti. RACK se posílá samostatně (od destination k source při jednosměrném přenosu) nebo připojený k datovému paketu (od source k destination nebo opačně při oboustranné komunikaci a při udržování aktivní cesty bez toku data paketů). Přeposláním packet RACK zvyšuje hopcount signatury.

4.4 Vytvoření mostu

Modul mostů se použije, pokud nastane chybový stav zapříčiněný výpadkem spojení při posílání datových paketů. Vytvoření mostu se koná namísto zneplatňování cesty a rozesílání RERR paketu. Pokud se most úspěšně vytvoří, provoz se ihned může obnovit. Pokud ne, proběhne klasická fáze zneplatnění cesty a rozeslání RERR paketu. Během tvorby se ukládají datové pakety směřující k destination opravované cesty do speciálního bufferu. Až se most vytvoří, pakety se mohou začít přeposílat. Pokud se most nevytvoří, pakety se zahazují.

Stavbu mostu zahájíme nastavením aktivní cesty k destination jako expired a vytvořením záznamu do bridge-tabulky, neboli tabulky stavěných mostů. Tabulka je indexovaná přes destination a obsahuje dále reference na bufferované pakety přijaté během opravy cesty a timeout pro vytvoření mostu. Pokud se most nepodaří vytvořit, bufferované pakety se zahodí. Cesta je definována jako v opravě, pokud je expired a existuje příslušný záznam v bridge-tabulce.

Při opravě cesty se tedy napřed vytvoří záznam v bridge-tabulce a poté se rozešle záplavovitě BREQ paket. Procedura je podobná jako u rozesílání RREQ paketu.

TTL se nastaví na malou hodnotu, typicky jako 3. Pro flood-ID se vygeneruje nové číslo, unikátní v rámci všech flood-ID různých typů paketů. Source údaje se zkopírují z routovací tabulky. Bridge-source je ID mobilu a bridge-source signatura je KNOWN signatura s inkrementovaným sekvenčním číslem a nulovým počtem standartních i mostních hopů.

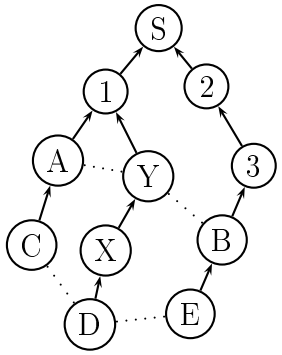
BREQ se opět šíří podobně jako RREQ, avšak za každý hop se inkrementuje bridgecount namísto hopcountu. Narazí-li BREQ na mobil s cestou s dostatečně velkou signaturou, odpoví paketem BREP. BREP se šíří zpět po vytvořených cestách k bridge-source a cestou opět zvyšuje bridgecount namísto hopcountu. Pokud BREP dorazí ve stanoveném časovém limitu k bridge-source, most je vytvořen, cesta updatována a označena jako aktivní a provoz se může obnovit. Paket BREP stejně jako RREP iniciuje zadávání mobilů do seznamu předchůdců.

Pokud paket BREP nedorazí v časovém limitu, vracíme se ke klasické proceduře chybového stavu. Cesta je prohlášena za neopravitelnou nastavením flagu "invalid" a rozešlou se RERR pakety tak, jak je definováno v 2.5. Na obrázku 4 znázorněno přemostění mobilu X, který "usnul" a přestal přeposílat pakety.

Nová cesta je v tomto případě o 1 hop delší. Na cestě je zachováno pravidlo rostoucí signatury. Cesta opačným směrem po přemostění už nemusí být shodná, viz odstavec konsolidace cesty.

4.5 Konsolidace cesty

Při přemostění čas od času vzniká situace, kdy cesta od source k destination není stejná jako cesta druhým směrem. Most se může postavit přes mobily, které kvůli dřívější znalosti cesty k source (tato znalost je pravděpodobná, neboť při počátečním floodingu RREQ paketů se cest k source vytvoří mnoho) při přijetí BREQ paketu cestu k source neupdatují, protože jejich cesta má menší signaturu. Uvažujme situaci na obrázku 5.



Obrázek 5: Vytvoření asymetrické cesty při tvorbě mostu přes vyřazený mobil X. Mobil A nepřesměruje svou aktivní cestu zpět přes Y.

Na cestě mezi source a destination vzniknul výpadek spojení kvůli mobilu X, který přestal přeposílat pakety. Šipkami jsou znázorněny cesty k source, jak byly vytvořeny po vlně RREQ paketů. Cesty k source na všech mobilech mají stejné sekvenční číslo.

Mobil Y zaznamená výpadek spojení, když se přes vypadlý uzel X pokouší přeposlat data. Nastoupí modul mostů a mobil Y rozesílá BREQ paket. Vlna BREQ paketů může postavit 2 mosty: přes mobily A a C nebo přes mobily B a E.

Na mostu přes B a E se zpáteční cesta vytváří obousměrně – uzel B si po přijetí BREQ vybere cestu k source přes Y, protože nabízí 2 standartní + 1 mostní hop, oproti původním 3 standartním. Podobně pro E. Pokud však zvolíme (stejně kvalitní) most přes A a C, pak rozdíl cest se projeví hned na uzlu A. Má si vybrat mezi novou cestou přes Y, nabízející 2 standartní + 1 mostní hop, oproti původním 2 standartním hopům. Mobil A proto nemůže přijmout cestu přes Y, dokud nová cesta nezvýší signaturu. Mobil A by dokonce cestu přes Y měl přirozeně odmítnout, protože přes tento mobil by mohla vést opravená cesta stejné délky jako původní.

Rozdílnost cest je důvod, kvůli kterému v BREQ paketu musíme rozlišovat source a bridge-source. Pokud by se BREP nedostal správně k místu výpadku, cesta by byla označena jako neopravitelná. Rozdílnost může být problém například při úsporném rozesílání RERR paketu.

Další problém vyplývá z rozložení signatur podél cesty. Modul mostů potřebuje pro svou činnost možnost prodloužit cestu. To je možné díky tomu, že mostní hopy mají menší váhu než standartní. To, že na cestě po opravě započítáváme mostní hopy, může být problém při další opravě, při stavbě "mostu na mostu". Problém rozdílnosti cest i problém mostních hopů na cestě lze řešit jednoduchým způsobem pomocí route acknowledgement paketů.

Source i destination po cestě pravidelně vysílají RACK pakety, samostatně nebo připojené k datovým paketům. Před vysláním koncovým uzlem RACK paket obdrží nově inkrementované sekvenční číslo a hopcount (a bridgecount) se nastaví na nulu. Tím, že RACK projde po aktivní cestě, přepíše bridgecount v routovacích tabulkách na hopcount (přepsání je korektní, protože jsme použili nové sekvenční číslo). Protože jsme updatovali záznamy o zpětné cestě po celé délce trasy, cesta tam i zpět již určitě vedou po stejných uzlech. Route acknowledgement pakety navíc neustále oddalují datum expirace cesty a tak můžeme díky opravám udržovat cestu dlouhodobě aktivní, aniž bychom se museli vracet k route discovery fázi.

4.6 Kombinace modulů

BREQ jsou flood-pakety, ale mají velmi omezený čas a proto nesmí být zařazeny do resend fáze flood redukce. Žádoucí úroveň floodingu pro BREQ by měla být větší nebo rovna úrovni pro RREQ pakety.

RACK pakety mohou být totožné jako pro modul route optimalizace (viz modul route-optimalizace).

4.7 Poznámky k využitelnosti modulu

Modul vykazuje zásadní zlepšení chování algoritmu AODV pro rozsáhlejší sítě, sítě s velkou mobilitou stanic a sítě s častými výpadky spojení. Umožňuje nastavovat delší čas pro expiraci cesty. Urychluje obnovení spojení po výpadku. Snižuje počet rozeslaných vln paketů RREQ. Na druhou stranu zvětšuje objem routovacích tabulek a prodlužuje aktivní cestu.

Další možné vylepšení tohoto modulu by bylo vytváření mostů po spolehlivých spojích ještě předtím, než by nastal skutečný výpadek spojení. Uvedená možnost zůstává objektem pro další zkoumání.

Modul mostů má parametry: timeout pro příjem paketů BREP a TTL pro vysílané BREQ pakety. Modul mostů také posunuje význam parametru životnosti cesty. U klasického AODV jsme byli limitováni pravděpodobností, že se cesta v krátkém čase "přetrhne". S modulem mostů jsme schopni cestu na mnoha místech opravit. I částečná informace na některých uzlech "po cestě" nám může pomoci nalézt spojení bez nutnosti nového širokého floodingu RREQ paketů.

Poznámka: Paket BREQ odpovídá paketu RREQ i po stránce zacházení s jednosměrnými linky. Pokud nedokážeme uzlu zpět přeposlat BREQ, zapíšeme ho na blacklist.

5 Modul route optimalizace

5.1 Idea optimalizace

Klasický algoritmus AODV zaručuje hledání nejkratší cesty v síti. Po určité době však nalezená cesta může kvůli pohybu mobilů přestat být nejkratší a nemusí být vůbec výhodná. Navíc při použití rozšiřujících modulů (mosty a flood-redukce) nejkratší cestu nemusíme mít zaručenou ani hned nebo krátce po vytvoření cesty.

Modul route optimalizace se snaží "vyhlazovat" cestu tam, kde by místo tří a více hopů stačily pouze dva, nebo namísto dvou a více jeden. Po vytvořené cestě se broadcastově posílají pakety tak, aby je slyšely okolní mobily a případně mohly nabídnout lepší cestu.

Modul route optimalizace používá nový druh paketu: RACK (taktéž používaný v modulu mostů, viz 4). Pakety se periodicky posílají po vytvořené aktivní cestě oběma směry. RACK posílá jeden koncový mobil (označme ho vysílací) druhému koncovému mobilu (označme přijímací) a nese vždy vysílací signaturu cesty k vysílacímu mobilu, v případě broadcastu i pomocnou přijímací signaturu k přijímacímu mobilu. RACK se posílá samostatně "adresovaným broadcastem" ve směru od destination k source, anebo připojený k datovému paketu ve směru od source k destination. Pokud máme obousměrný datový tok, můžeme RACK adresovaným broadcastem posílat oběma směry.

Adresovaný broadcast znamená, že paket je sice broadcastován, avšak jedna adresa je v paketu zdůrazněna jako příjemce a pouze tento konkrétní mobil má právo RACK přeposlat. RACK se adresuje tak, aby se přeposílal právě pouze po použité aktivní cestě k přijímacímu mobilu. Okolní mobily RACK pakety poslouchají a mají-li možnost, nabídnou výhodnější spojení

5.2 Přeposílání optimalizačního paketu

Před odesláním RACK paketu vysílací inkrementuje své sekvenční číslo a zadá ho do vysílací signatury a hopcount nastaví na nulu. Při přeposlání se zvyšuje hopcount vysílací signatury. Díky novému sekvenčnímu číslu RACK updatuje tabulky po celé délce cesty. Při broadcastu, přijímací signaturu mobil vždy jen opíše ze své routovací tabulky podle cesty k přijímacímu mobilu, a dále podle nexthopu této cesty nastaví adresu broadcastu. Všechny naslouchající mobily mají tedy přehled o údajích v routovacích tabulkách tam i zpět pro všechny odesílatele zachycených broadcastovaných RACK paketů.

Naslouchající mobily, jimž není broadcastovaný paket adresován, se rozhodují podle vysílací signatury. Pokud je signatura větší, než mají u odpovídajícího záznamu v routovací tabulce, pak routovací tabulku updatují. Pokud mají obě signatury stejné sekvenční číslo a záznam v tabulce má minimálně o 2 menší hopcount, pak nabídnou lepší cestu.

Přesměrování cesty proběhne zasláním dvou RREP paketů adresovaných:

- mobilu uvedenému v routovací tabulce jako nexthop na cestě k vysílacímu mobilu
- odesílateli zachyceného RACK paketu

Přijímací signaturu využijeme pro update routovací tabulky. (Může se za velmi specifických okolností stát, že update nemůže kvůli signaturám proběhnout. V tom případě proces optimalizace zrušíme. Po dvojici RACK paketů, které projdou cestou oběma směry by se však situace neměla opakovat). Signatury RREP paketů opíšeme obě z updatované routovací tabulky.

U RACK paketů, respektive pro modul route optimalizace se volí parametr periody odesílání RACK paketů, připojování k datovým paketům a požadavky na acknowledgement při unicastovém přeposílání libovolného paketu.

6 Simulace

6.1 MOBNET

Pro účely testování, analýzy a vizualizace algoritmů routování v mobilních ad-hoc sítích byla vytvořena a použita aplikace MOBNET (podrobně je aplikace zdokumentovaná v [7]), analýzy proběhly s verzí 1.5.

MOBNET je dynamická knihovna (dll), která se přilinkuje k programu protokolu a zajistí funkčnost a uživatelské rozhraní celého prostředí. Programátor protokolu má k dispozici čistě abstraktní třídu, ke které musí doimplementovat virtuální metody. V těchto metodách definuje inicializaci datových struktur, chování mobilů, zadávání datových přenosů a textové výstupy (libovolný text: stav tabulek, přeposlané pakety, analýzy). Výsledná aplikace provádí diskrétní simulaci (po krocích) a zobrazuje. Umožňuje také základní interakci s uživatelem pro účely testování, tj. zadávat datové přenosy a množství automaticky generovaných přenosů, spouštět, pozastavovat a krokovat simulaci, selectovat jednotlivé mobily (kvůli detailnímu textovému výstupu) a spouštět UserFunkci s volitelným významem podle implementace.

Rozhraní MOBNETu bylo uzpůsobeno objektové implementaci algoritmů a zaměřuje se na srovnávací testování a analýzu. V rámci implementace v aplikaci MOBNET můžeme hovořit o další vrstvě, tzv. analytické vrstvě. Analytická vrstva se nachází těsně nad simulovanou linkovou vrstvou a jejím úkolem je

- shromažďovat údaje o odeslaných paketech pro vytváření statistik
- upravovat vlastnosti vizualizace (barva, zobrazitelnost)
- shromažďovat údaje o datovém toku a úspěšně přijatých datových paketech

MOBNET naopak nerozebírá linkovou a fyzickou vrstvu. Díky tomu program zpracovává čtvrtinu paketů – pouze data, namísto RTS/CTS/Data/ACK. Vliv linkové vrstvy na omezení kapacity sítě popisuje [9] a s určitou tolerancí lze započíst dodatečně.

Celý projekt i algoritmy routování jsou napsány v jazyce C++ a využívají grafickou knihovnu SDL. Knihovna SDL je sama o sobě cross-platform, avšak projekt je vytvořen pro OS Windows. MOBNET byl vytvořen v rámci ročníkového projektu.

6.2 Simulační model

Simulace se odehrává na mapě. Mapa se skládá ze sektorů a sítě cest. Každý sektor a každá cesta je určitého druhu (druh terénu, typ cesty) a podle druhu má odvozeny především vlastnosti:

- koeficient rychlosti (rychlost mobilu v terénu určena náhodně v rozsahu přenásobeném tímto koeficientem)
- koeficient důležitosti (ovlivňuje pravděpodobnost vybrání sektoru a cesty jako cíl pohybu mobilu)

Mobil se pohybuje rovnoměrně přímočaře k cíli cesty. Cíl vybírá podle důležitosti cesty a okolních sektorů. Startuje-li ze sektoru, vybírá buďto náhodný bod v nějakém

blízkém sektoru (dosah 2 sektory) nebo nějaký blízký počátek cesty. Rozhoduje-li se na uzlu cest, vybírá cíl jako druhý konec některé přilehlých cest, anebo zvolí cesty úplně opustit a jít k náhodnému bodu přiléhajícího sektoru. Důležitost objektů v důsledku ovlivňuje hustotu mobilů v sektoru, respektive provoz na cestách.

Simulaci lze orientačně přidělit rozměry – mapa je velký čtverec se stranou 1,5 km, sektor je malý čtverec se stranou 50 m, krok má délku 0,1 s a mobily se pohybují rychlostí okolo 0,5 m/s (v budovách), okolo 10 m/s (na cestách) a okolo 2,5 m/s ve volném terénu. Pohyb však násobíme pohybovou konstantou pro zkoumání chování sítě při různých podmínkách. Dosah signálu je v rámci rozměrů mapy nastavován na hodnotu okolo 100m, mobily mají stejný dosah (nesimulujeme tzv. jednosměrné linky). Životnost cesty je 8 vteřin.

Mobnet umožňuje diskrétní simulaci, tedy simulaci po krocích. Zpráva odeslaná v jednom kroku bude zpracována až v dalším kroku. Zprávy neinterferují a posílají se spolehlivě. Zpráva se úspěšně přenese právě tehdy, když je cílový mobil aktivní a v dostatečné blízkosti od odesílatele. Mobil nedokáže modifikovat sílu signálu ani signál směrovat. Výpadky spojení nastávají, pokud se od sebe vedlejší mobily příliš vzdálí, nebo pokud mobil přestane být aktivní (stav sleep). Interference lze společně se zahlcením pozorovat podle počtu přijatých paketů nebo přes úroveň hluku z okolí.

Mobil má každý krok určitou pravděpodobnost, že se pokusí o spojení s náhodným dalším mobilem. Spojení má charakter jednosměrného hovoru. Je náhodně nastaven počet datových paketů na přeposlání, od 1 až k stanovenému limitu, přičemž tento počet se nezvětšuje při výpadcích- neuvažujeme přeposílání ztracených dat. Nezávisle analyzujeme kvalitu spojení, tedy kolik paketů se přeposílá úspěšně, a řídící provoz spojený s posíláním těchto dat.

Dále má mobil pro každý tah stanovenou pravděpodobnost "usnutí", neboli dočasné deaktivace. V takovém případě zahodí přijaté zprávy a nějakou dobu vůbec neparticipuje na provozu sítě.

6.3 Poznámky k implementaci routovacího algoritmu

V práci je popsáno mnoho variant pro detaily chování algoritmu. Varianta lze zvolit tak, aby vyhovovala danému rozšíření a vlastnostem simulačního prostředí. Všechny testy proběhly na implementacích s následujícími parametry:

- úsporné rozesílání RERR paketů, tj. unicastově k source
- využíváme služeb přenosového rozhraní, který potvrzuje přijetí každého unicastového přenosu.
- prostředí nesimuluje jednosměrné spoje, nepoužívají se tudíž blacklisty
- broadcast signál se přijímá pouze při dostatečné síle signálu, dosah broadcastu a unicastu nastavován v poměru 1:1,1. Uváděn bude vždy dosah broadcastu.
- nijak neimplementujeme bezpečnost

6.4 Výsledky simulace

U algoritmu nás zajímá, jak moc "dobře" umí přeposílat datové pakety v různých podmínkách. Nastavení sledovaných veličin a sledované výsledky vychází ze specifikace [3]. Vzhledem k tomu, že neuvažujeme vliv linkové vrstvy na velikost rámců, omezíme se na sledování po paketech. Přímo měříme tyto veličiny:

all packets. Celkový počet paketů, každý paket se samozřejmě započítává tolikrát, kolikrát byl přeposlán.

all data. Počet přeposlání datových paketů, počítají se i přeposlání datového paketu, který nedorazí k cíli

delivered. Počet datových paketů přijatých destination mobilem

reliability, data delivery [%]. Spolehlivost. Pouze pro ukončenou relaci, je přesně podíl přijatých a odeslaných datových paketů. Jedná se o "hrubou" spolehlivost, bez podpory vyšších vrstev, které by nahrazovaly ztracené pakety.

neighbors. Konektivita – průměrný počet sousedů.

A odvozujeme následující veličiny:

packet overhead. Počet všech odeslaných paketů na 1 přijatý datový paket, $\frac{all\,packets}{delivered}$.

data hops. Průměrný počet přeposlání datového paketu na jeden doručený paket, tedy $\frac{all\,data}{delivered}$. Pokud není extrémní ztrátovost, odpovídá průměrné délce aktivních cest.

distributed overhead. Rozložená zátěž. Průměrný počet zpráv, který se vygeneruje na každém mobilu při přeposlání 1 datového paketu od source k destination. Vzorec $\frac{all\,packets}{delivered * mobils}$.

throughput. Propustnost sítě. Odhadneme zhruba jako $\frac{capacity}{distributed\,overhead * connectivity}$. Problém propustnosti je však složitější a je rozebírán například v [9]. Omezme se proto na hodnocení předchozí veličiny distributed overhead.

Simulace probíhala v poměrně krátkých časových úsecích (150 až 200 sekund) a proto je zatížena určitou chybou. Slouží nejlépe jako srovnání efektivity modulů z různých stránek pro různě parametrizované prostředí.

V první fázi (obr. 6) testujeme klasický algoritmus AODV a moduly flood redukce pro různě nastavené parametry (žádoucí úrovně a minimální úrovně floodingu) na různě rozsáhlých sítích. Snažíme se přitom zachovat hustotu mobilů, takže průměrný počet sousedů se pohybuje okolo 40, podle terénu mapy jsou oblasti s vyšší nebo nižší hustotou. Vidíme, jak rostou nároky na počet rozeslaných paketů pro větší sítě. Zajímavým pozorováním je, že pro velké sítě modul flood redukce ušetří o málo *větší*

procento paketů. Je tomu tak proto, že flooding produkuje množství paketů rostoucí lineárně s počtem uzlů sítě, avšak délka cesty pouze s odmocninou počtu mobilů, jak bylo řečeno v sekci 3. Pro velké sítě je proto primárním cílem ušetřit při floodingu, i za cenu prodloužení cesty. Spolehlivost se pro sítě s dostatečným stupněm konektivity pro různé parametry redukce nemění.

Zadruhé testujeme vliv hustoty mobilů na výkon AODV a modulu flood redukce (obr. 7). Prokazatelný pozitivní vliv začíná u množství 1000 – 2000 mobilů, u průměrného stupně konektivity okolo 18 uzlů. Pro velmi malý počet uzlů narážíme na problém nesouvislé sítě. Hopcount a počet paketů dokonce klesne, protože se pakety nedostanou za hranice komponenty souvislosti. Důsledkem modulu by mělo být i rovnoměrnější využívání media.

Třetí sada simulací (obr. 8) testuje všechny moduly. Modul flood redukce je vždy zapnutý na úrovně 14 a 11. Porovnáváme klasické AODV, modul redukce, modul redukce a mostů, modul redukce a mostů a route optimalizace. Parametry mapy nastavujeme tak, aby se měnil počet mobilů a hustota zůstávala stejná. Výsledný efekt je ekvivalentní zvětšování plochy a proporcionálně počtu mobilů. Pro velké sítě dosahujeme výrazně lepších výsledků s modulem mostů. Přestože modul route optimalizace nepřináší výrazné snížení počtu všech poslaných paketů na jeden přijatý datový paket, graf datového hopcountu ukazuje, že přinejmenším snižuje počet přeposílaných datových paketů.

Ve čtvrté fázi podrobujeme moduly testům prostředí podle různého koeficientu pohybu uzlů. Výsledky na obr. 9. Opět se ukazuje, že modul mostů zřetelně zmenšuje řídící provoz za ztížených podmínek. Ve velmi nestálém prostředí se stává čím dál častěji, že v jednom okamžiku neexistuje spojitá aktivní cesta mezi source a destination. Pakety BREQ neustále vyhledávají nejkratší cestu vpřed a data se přeposílají vždy trochu blíže destination. Jako problém se pak ukazuje, že (neopravované) pakety RREQ se často vůbec nemusí dostat zpět k source.

Poslední fáze zobrazená na obr. 10 srovnává počet paketů na přenesený datový paket pro různé kombinace modulů. Simulace se odehrává na mapě s 4000 mobily a konektivitou okolo 17 sousedů. Část práce do budoucna by se měla ubírat i k hledání vhodně zvolených parametrů vzhledem k charakteristice sítě.

7 Závěr

Překážkou širokého použití mobilních ad hoc sítích je velký nárůst řídícího provozu a chybovosti se zvětšováním velikosti sítě. V rámci této práce byly navrhnuty a prezentovány tři moduly, které upravují chování algoritmu tak, aby byl schopen efektivně fungovat i v prostředí, které je svými charakteristikami pro ad-hoc networking velmi nepříznivé. Zejména překonáváme problémy s nahuštěním mobilů, problémy s častými výpadky, velkou mobilitou uzlů a cestami skrz rozsáhlou síť. Často pak dosahujeme mnohonásobného snížení počtu poslaných paketů. Osvědčily se zejména modul mostů a modul flood redukce.

Reference

[1] Abdulrahman H. Altalhi and Golden G. Richard III. Virtual paths routing: A highly dynamic routing protocol for ad hoc wireless networks. *PerCom Workshops*, 2004.

[2] K. Bhargavan, D. Obradovic, and C. Gunter. Formal verification of standards for distance vector routing protocols. *UPenn Tech Report*, 1999.

[3] S. Corson and J. Macker. Mobile ad hoc networking (manet) routing protocol performance issues and evaluation criteria. *RFC 2501*, Jan 1999.

[4] X. Hong, K. Xu, and M. Gerla. Scalable routing protocols for mobile ad hoc networks. *IEEE Network Magazine*, Jul–Aug 2002.

[5] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol. *internet draft*, 2004.

[6] Brad N. Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. *MobiCom '00*, 2000.

[7] M. Káldy. Mobnet dokumantace. *dokumantace k ročníkovému projektu*, 2006.

[8] L. Kučera. Low degree connectivity in ad-hoc networks. *Proceedings of ESA*, 2005.

[9] Jinyang Li, Charles Blake, Douglas S. J. De Couto, Hu Imm Lee, and Robert Morris. Capacity of ad hoc wireless networks. *Mobile Computing and Networking*, pages 61–69, 2001.

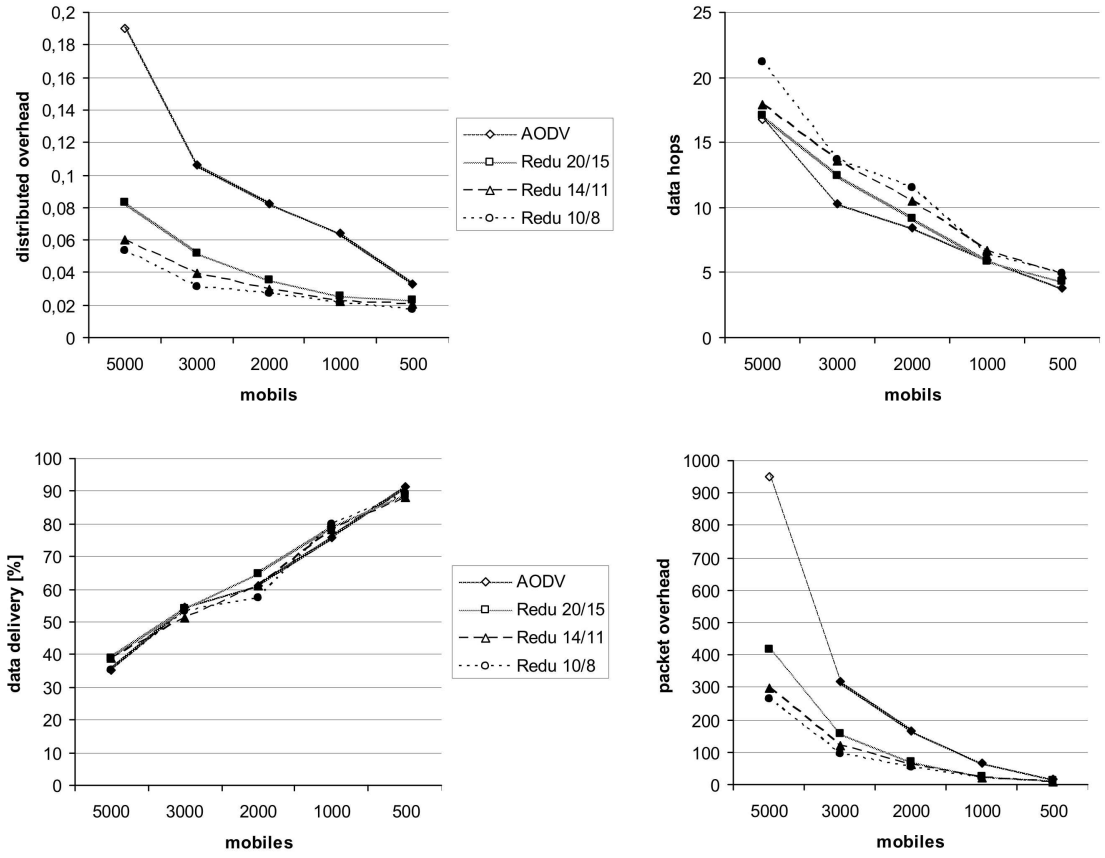
[10] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. *RFC 3561*, Jul 2003.

[11] I. Chlamtac S. Basagni, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility (dream). *Mobicom '98*, 1998.

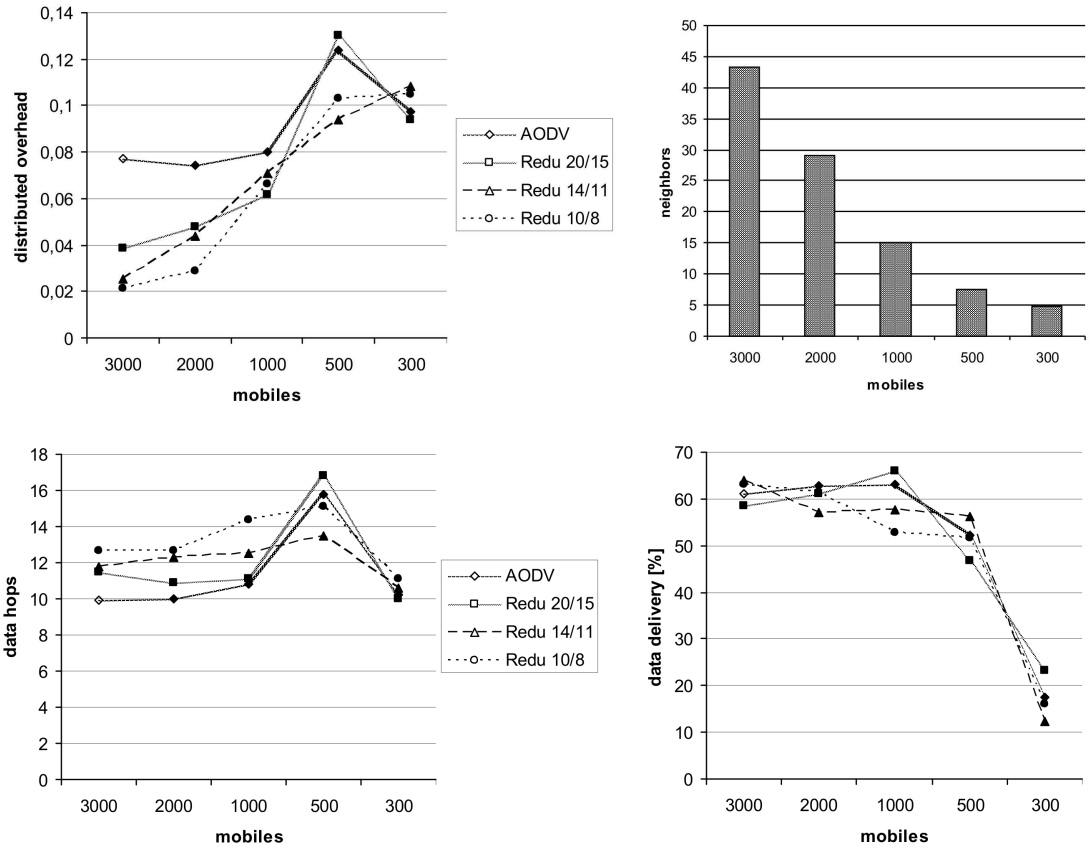
[12] S. Sanyal, A. Abraham, D. Gada, R. Gogri, P. Rathod, Z. Dedhia, and N. Mody. Security scheme for distributed dos in mobile ad hoc networks. http://falklands.globat.com/~softcomputing.net/iwdc-manet.pdf.

[13] H. Zimmermann. Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communication*, 1980.

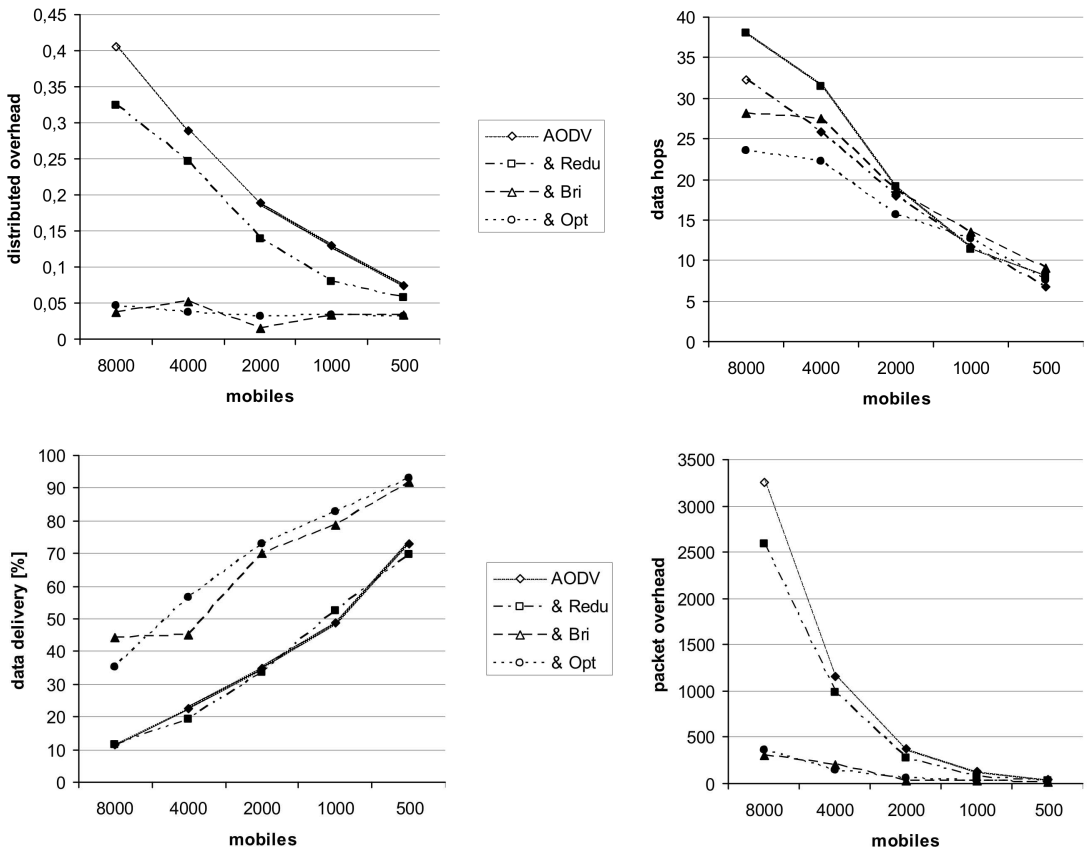
8 Příloha: Grafy



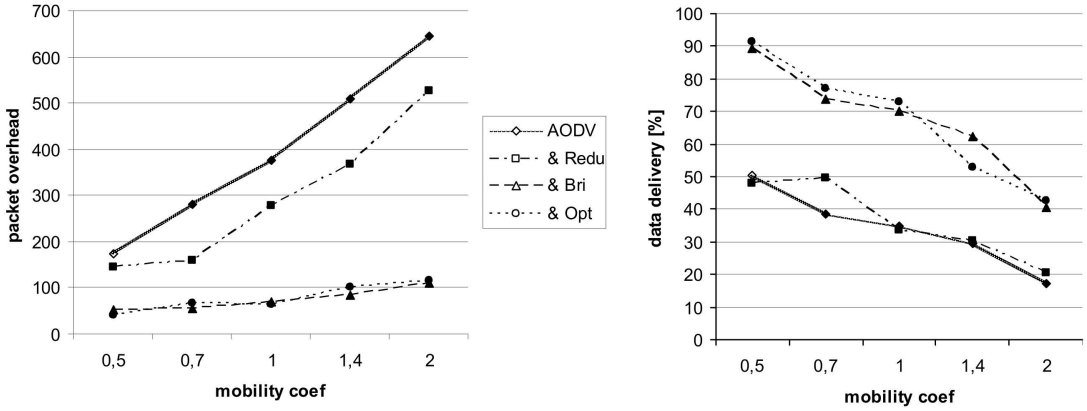
Obrázek 6: Průměrná rozložená zátěž, data hopcount, spolehlivost a celkový výkon na 1 úspěšně přeposlaný datový paket. Analýza klasického AODV a rozšíření flood-reduction (pro různě nastavené žádoucí a minimální úrovně floodingu) podle velikosti sítě, tak aby počet sousedů se udržoval na konstantní úrovni okolo 40 mobilů. Při takto vysoké úrovni konektivity se zřetelně pozitivně projevuje efekt flood-redukce.



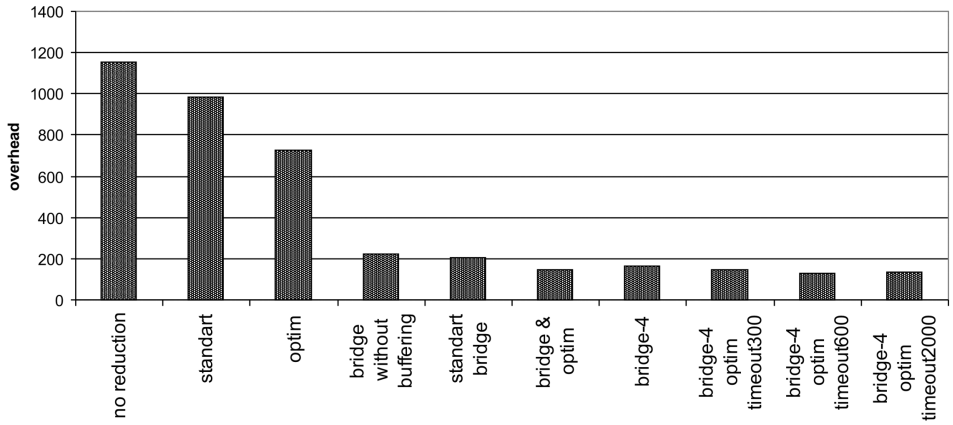
Obrázek 7: Vyhodnocení klasického AODV a rozšíření flood-reduction podle různého množství mobilů na konstantní ploše 1,5 km x 1,5 km a dosahu signálu 100 m. Vlevo nahoře rozložená zátěž na přeposlaný paket od source k destination. Vpravo nahoře průměrný počet sousedů. Při počtu sousedů pod 6 se síť již stává nesouvislou. Vlevo dole datový hopcount. U malého množství mobilů opět klesá, protože se uskutečňují pouze lokální přenosy. Vpravo dole spolehlivost spojení. Pro malé množství mobilů prudce klesá.



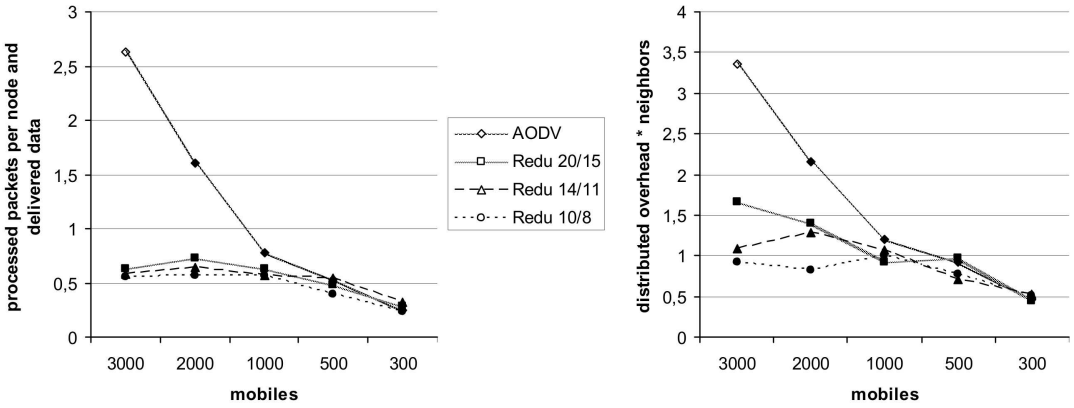
Obrázek 8: Vyhodnocení AODV a postupně přidávaných modulů pro různě rozsáhlé sítě. Průměrný počet sousedů je vždy okolo 17. Grafy zobrazují rozloženou zátěž, data hopcount, spolehlivost a celkový výkon na 1 úspěšně přeposlaný datový paket. Vliv modulu mostů je pro velké sítě zřetelně vidět, jak ve spolehlivosti, tak v množství paketů. Pro mosty je paradoxně hopcount nižší, přestože prodlužují trasu, po které se musí pakety přeposílat. Vypočítaný hopcount se ale významně snižuje, protože jen minimum paketů se zahazuje. Efekt route optimalizace není zřetelný.



Obrázek 9: Vyhodnocování AODV a postupně přidávaných modulů pro různou mobilitu uzlů. Parametry simulace jsou kromě pohybu konstantní. Počet mobilů je 2000, průměrný počet sousedů 17. Algoritmus s modulem mostů vykazuje ve větších rychlostech mnohem lepší výsledky.



Obrázek 10: Vyhodnocení různých kombinací modulů pro mapu s 4000 mobily a konektivitou 17 sousedů. Mimo jiné byly testovány varianty modulu mostů, který definuje velmi dlouhý route timeout a prodlužuje dosah mostů na 4 hopy (bridge-4). V sítích od několika tisíců uzlů výše přináší malé zlepšení výkonu, v menších sítích se však tyto modifikace neosvědčily.



Obrázek 11: Vyhodnocení počtu zpracovaných paketů na přijatý paket a mobil pro různě nastavené parametry flood redukce (žádoucí a maximální úroveň floodingu), v závislosti na počtu mobilů na konstantně velké ploše 1,5 km x 1,5 km, dosah signálu je 100 m. Zatímco distributed overhead v předchozích grafech odpovídal výkonu, počet zpracovaných paketů odpovídá zahlcení. Rozdílně se počítají pakety, které se pouze "zaslechnou", avšak dále se nepřeposílají (pakety patřící do již zpracované vlny RREQů, flood-redukované pakety apod.) Takové pakety se v tomto případě započítávají.

Dále je tu problém interferencí. Na dlouhé vzdálenosti závisí na výkonu sítě. Na krátké vzdálenosti by k interferencím nemělo docházet – díky přístupové metodě k médiu implementované na linkové vrstvě. Vytíženost média (z které vyplývá množství kolizí) odhadneme jako distributed overhead přenásobený počtem sousedů. Výsledek je uveden na grafu napravo. Téma vzájemného rušení v mobilních ad hoc sítích je však složitější a pro vyhodnocení se odkažme na existující analýzu [9]. Pokud srovnáme grafy nahoře, vidíme, že obě veličiny vychází velmi podobně.