Fakulta matematiky, fyziky a informatiky
Univerzita Komenského, Bratislava
Katedra Informatiky

Škálovatelné smerovanie v Ad-Hoc sieťach

diplomová práca

Autor: Ján Oravec
Vedúci dipl.práce: RNDr. Rastislav Královič, PhD.
Bratislava, Máj 2006

Čestne prehlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej literatúry.

Ján Oravec

Ďakujem svojmu diplomovému vedúcemu RNDr. Rastislavovi Královičovi, PhD. za cenné rady a pripomienky.

Ďakujem svojej rodine a priateľom za podporu pri písaní tejto práce.

Abstrakt

V tejto diplomovej práci sme navrhli škálovatelný smerovací protokol *SHAR* pre Ad-Hoc siete. Pamäťové nároky stanice sme znížili vybudovaním hierarchickej štruktúry v sieti. Povolením súčasného členstva stanice vo viacerých klastroch na rovnakej úrovni hierarchie sme umožnili lenivú voľbu klastrov, ktorá výrazne znižuje komunikačné nároky stanice. Obmedzením funkcie šéfov klastrov na tvorbu hierarchie zlepšujeme smerovanie dát, ktoré nie sú na rozdiel od podobných protokolov smerované cez šéfov klastrov v hierarchii. Rýchlosť konvergencie siete sme znížili vhodnou kombináciou prvkov algoritmov *Link State* a *Distance Vector*.

Kľúčové slová: Ad-Hoc siete, MANET, smerovací protokol, smerovanie dát

Obsah

Kapitola 1

Úvod

Prvé Ad-Hoc siete boli vybudované pred viac ako 30 rokmi organizáciou DARPA[1]. Tieto siete pozostávali z komunikačných staníc vzájomne prepojených rádiovými spojeniami a nazývali sa *packet radio* siete [Kah77]. Rast počtu pripojených staníc a nestabilita rádiových spojov podnietili vývoj protokolov zabezpečujúcich smerovanie dát v sieťach s dynamicky meniacou sa topológiou. Neskôr sa pridala požiadavka mobility pripojených staníc – potreba zabezpečenia neprerušenej komunikácie počas pohybu stanice v sieti. Takéto siete sa častokrát v literatúre nazývajú *MANET*[2] [Bak02].

Minimálna potreba konfigurácie staníc umožňuje rýchle nasadenie Ad-Hoc sietí, čím sa stávajú vhodným komunikačným prostriedkom v núdzových situáciách, ako sú prírodné alebo človekom spôsobené katastrofy, ozbrojené konflikty, a podobne. Taktiež sú vhodné na miestach s chýbajúcou infraštruktúrou, kde sa dajú efektívne použiť pri výuke. Ako príklad možno spomenúť projekt OLPC[3], ktorého cieľom je vyvinúť lacný laptop pre školy v chudobných krajinách. Tieto laptopy budú navzájom prepojené prostredníctvom Ad-Hoc siete. Ak sa vyriešia problémy škálovatelnosti v Ad-Hoc sieťach, je možné v budúcnosti možno očakávať rozsiahle verejné Ad-Hoc siete tvorené mobilnými zariadeniami.

Mnoho vedcov sa od vzniku prvých *packet radio* sietí venovalo problému smerovania v Ad-Hoc sieťach. Autori jednotlivých článkov častokrát využívali rôzne predpoklady týkajúce sa najmä fyzickej vrstvy, nad ktorou bola sieť vytvorená. Najčastejšie používaný model bol motivovaný rádiovou sieťou – stanice sa nachádzajú v dvojrozmernom priestore, pričom spojenie medzi dvoma stanicami existuje práve vtedy, ak sú vzdialené dostatočne blízko od seba.

[1] Defense Advanced Research Projects Agency
[2] Mobile Ad-Hoc NETwork
[3] One Laptop per Child

Tento model dobre aproximuje realitu v rádiových sieťach s diaľkovými spojeniami. Avšak v prostredí súčasných veľkomiest je tento model veľmi skreslený – dvojrozmerný priestor by bolo nutné nahradiť trojrozmerným s veľkým množstvom prekážok, na ktorých dochádza k prudkému útlmu rádiového signálu. Preto v tejto práci nebudeme predpokladať žiadne topologické vlastnosti siete a budeme sa snažiť navrhnúť smerovací protokol pre všeobecnú Ad-Hoc sieť – teda takú, ktorej spojenia staníc tvoria ľubovoľný graf. Budeme však predpokladať, že sieťová vrstva pod smerovacím protokolom nás bude pravdivo informovať o novovytvorených a zrušených spojeniach a týmito spojeniami bude spoľahlivo doručovať správy – nezmenené, v pôvodnom poradí a bez straty (s výnimkou správ poslaných tesne pred uzavretím spojenia druhou stranou).

Existujúce smerovacie protokoly pre Ad-Hoc siete majú vážny problém so škálovatelnosťou – so zväčšujúcim sa počtom staníc rýchlo rastú pamäťové nároky, počet prenesených správ prípadne prudko vzrastie čas konvergencie siete a sieť je pri častých zmenách topológie úplne nefunkčná. Týmto problémom sa budeme snažiť v našom návrhu protokolu vyhnúť.

Táto práca je rozdelená do nasledujúcich kapitol:

Kapitola 2 – Základné definície a pojmy formálne definuje použitý model Ad-Hoc siete a objasňuje často používané pojmy.

Kapitola 3 – Prehľad problematiky bližšie uvádza čitateľa do problematiky smerovania v Ad-Hoc sieťach. V tejto kapitole je možné nájsť rozdelenie smerovacích protokolov, algoritmy v nich používané a popis protokolu *HSR*[4] [PGHC99].

Kapitoly 4 až 9 – Protokol Scalable Hierarchical Ad-Hoc Routing sa zaoberajú návrhom protokolu založenom na budovaní hierarchickej štruktúry v sieti. Tieto kapitoly prinášajú nové výsledky v škálovatelnosti smerovacích protokolov v Ad-Hoc sieťach.

Kapitola 10 – Záver zhodnocuje dosiahnuté výsledky a naznačuje možné smerovanie ďalšieho výskumu.

[4]Hierarchical State Routing Protocol

Kapitola 2

Základné definície a pojmy

Táto kapitola popisuje definície a pojmy často používané v tejto práci.

Ad-Hoc sieť – sieť pozostávajúca zo staníc prepojených spojeniami. Túto sieť je možné v každom momente reprezentovať neorientovaným grafom, ktorého vrcholy reprezentujú stanice a hrany reprezentujú spojenia. Na graf Ad-Hoc siete nie sú kladené žiadne ďalšie požiadavky. Stanice a spojenia Ad-Hoc siete môžu v čase pribúdať a miznúť.

Stanica – implementuje smerovací protokol Ad-Hoc siete. Každá stanica má pridelený unikátny identifikátor a vie o všetkých spojeniach so susednými stanicami.

Identifikátor stanice – číslo pridelené centrálnou autoritou. Každá stanica má unikátny identifikátor.

Spojenie – obojsmerný komunikačný kanál medzi dvomi stanicami A a B. Každé spojenie má vzdialenosť – kladnú cenu prenesenia správy medzi stanicami. Táto cena môže byť pre rozdielna pre jednotlivé smery. Stanica A pozná cenu prenesenia správy zo stanice A ku stanici B. Cena spojenia je väčšinou definovaná ako latencia spojenia – dĺžka trvania prenesenia informácie medzi dvomi stanicami. Cena sa v čase môže meniť. Väčšinou sa smerovaciemu protokolu propaguje informácia o zmene ceny spojenia len keď je dosiahnutá dostatočne veľká zmena.

Systém – komunikačná jednotka, ktorá využíva smerovací protokol pre komunikáciu v Ad-Hoc sieti. Obsahuje subsystém zabezpečujúci vytváranie a rušenie spojení a spoľahlivý prenos správ týmito spojeniami. Zatiaľ, čo pod pojmom *systém* chápeme fyzickú jednotku, pod pojmom *stanica* chápeme len jej časť implementujúcu smerovací protokol.

Cesta medzi z A do stanice B – postupnosť susedných staníc začínajúcej sa v stanici A a končiacej v stanici B.

Cena cesty – súčet ceny orientovaných spojení, ktorými je cesta tvorená.

Virtuálne spojenie – spojenie medzi dvomi stanicami, ktoré sa javí ako priame spojenie, ale je tvorené nejakou cestou medzi týmito dvomi stanicami.

Správa – informácia prenášaná medzi systémom a stanicou. Táto definícia zahrňuje informácie posielané cez spojenia medzi dvomi susednými stanicami, keďže samotné doručenie nezabezpečuje stanica, ale systém, ktorému je správa poslaná. Správy slúžia na oznamovanie o vytvorených spojeniach stanici, na výmenu smerovacích informácií medzi dvoma stanicami a na doručovanie dát.

Dáta – informácia prenášaná medzi dvomi systémami. Smerovací protokol zabezpečuje doručenie dát použitím správ posielaných medzi stanicami.

Broadcasting – metóda posielania správ všetkým staniciam v sieti.

Smerovací protokol – protokol zabezpečujúci smerovanie dát medzi dvomi systémami. Protokol si vymieňa smerovacie informácie prostredníctvom správ. Na základe týchto informácií doručuje dáta stanici s požadovaným identifikátorom. Tieto dáta sú odovzdané systému na spracovanie.

Konvergencia siete – adaptácia smerovacieho protokolu na zmeny v topológii. Počas konvergencie si stanice správami vymieňajú smerovacie informácie.

Hierarchická topológia – systém organizácie siete pozostávajúci z viacerých úrovní. Najnižšia úroveň pozostáva zo staníc. Každá ďalšia úroveň je tvorená množinou klastrov, pričom každý objekt (stanica/klaster) nižšej úrovne patrí do práve jedného klastra. Najvyššia úroveň pozostáva z jedného klastra.

Klaster úrovne L – je tvorený objektmi na nižšej úrovni hierarchie. V prípade, že $L = 0$, nižšia úroveň pozostáva zo staníc. V opačnom prípade je nižšia úroveň tvorená klastrami.

Šéf klastra – význačný člen klastra, ktorý organizuje klaster. Definícia sa častokrát poníma v tranzitívnom tvare, teda, že šéfom klastra na úrovni L je stanica, ktorá je šéfom šéfa tohto klastra na úrovni $L - 1$.

Hodnosť stanice – číslo, udávajúce počet úrovní, v ktorých je táto stanica šéfom klastra.

Hierarchický identifikátor stanice – postupnosť identifikátorov šéfov klastrov po ceste zo stanice hore v hierarchii.

Domáci agent – stanica, ktorá uchováva informáciu o hierarchickom identifikátore danej stanice.

Kapitola 3

Prehľad problematiky

Cieľom tejto kapitoly je oboznámiť čitateľa so základnými prístupmi v problematike smerovania v Ad-Hoc sieťach.

3.1 Kritériá smerovacích protokolov

Keďže stanice komunikujúce v Ad-Hoc sieťach nemajú vopred známu topológiu siete, na to, aby mohli smerovať dáta, musia túto topológiu určitým spôsobom spoznať. Základnou myšlienkou každého smerovacieho protokolu je výmena smerovacích informácií medzi susednými stanicami. Rôzne protokoly sa líšia v type informácií prenášaných pri zmene topológie, druhu pamätaných informácií zúčastnenou stanicou a v tom, akú činnosť je treba vykonať na vytvorenie spojenia s danou cieľovou stanicou.

Každý protokol je možné opísať niekoľkými jeho vlastnosťami:

Pamäťové nároky stanice - definuje závislosť veľkosti potrebných pamätaných smerovacích informácií od počtu staníc pripojených v sieti. Aby sme dosiahli vysokú škálovatelnosť protokolu, budeme vyžadovať rádovo nižšie ako lineárne pamäťové požiadavky.

Optimálnosť voľby cesty k cieľovej stanici - budeme skúmať závislosť ceny cesty použitej protokolom v závislosti od ceny najlacnejšej cesty.

Konvergencia siete - keďže zmeny topológie v Ad-Hoc sieťach môžu prebiehať často, je dôležité dosiahnuť dobré časy konvergencie siete. Taktiež sa budeme venovať vplyvom konvergencie na prebiehajúcu komunikáciu.

Komunikačná efektivita - budeme pozorovať závislosť počtu prenesených správ potrebných na výmenu smerovacích informácií od veľkosti a topológie siete.

Oneskorenie pri vytvorení spojenia - čas potrebný na získanie cesty k požadovanej stanici.

3.2 Delenie smerovacích protokolov

Podľa týchto vlastností delíme smerovacie protokoly v Ad-Hoc sieťach do dvoch základných skupín:

reaktívne - zisťujú a získavajú cestu pred vytvorením spojenia

proaktívne - udržujú smerovaciu tabuľku používanú na smerovanie dát

3.2.1 Reaktívne protokoly

Reaktívne protokoly si nepamätajú smerovacie informácie o cieľových staniciach, s ktorými neprebieha žiadna komunikácia. Preto sú ich pamäťové a komunikačné požiadavky na udržiavanie smerovacích informácií veľmi nízke. Každá požiadavka na vytvorenie komunikácie však vyžaduje broadcastovanie správy obsahujúcej požiadavku na nájdenie cesty medzi zdrojovou a cieľovou stanicou. To spôsobuje oneskorenie pri vytváraní spojenia a vo veľkých sieťach ich zahlcovanie. Tieto vlastnosti reaktívnych protokolov výrazne znižujú ich škálovatelnosť.

3.2.2 Proaktívne protokoly

Aktívnym udržiavaním smerovacích informácií o každej cieľovej stanici nemajú proaktívne protokoly oneskorenie pri vytvorení spojenia. Nutnosť udržiavania informácií o staniciach, s ktorými sa nekomunikuje však zvyšuje pamäťové a komunikačné požiadavky. Ďalším problémom je oneskorená propagácia zmien topológie spôsobujúca dlhú dobu konvergencie. Proaktívne protokoly preto tiež nie sú vhodné na použitie vo veľkých Ad-Hoc sieťach.

3.3 Smerovacie algoritmy

Pred tým, ako budeme hľadať východiská z problémov reaktívnych a proaktívnych protokolov si ešte popíšme algoritmy, na ktorých je založená väčšina

proaktívnych protokolov. Tieto algoritmy nám poslúžia ako základ ďalších protokolov.

3.3.1 Algoritmus *Link State (LS)*

Myšlienka algoritmu *LS* je založená na prenášaní stavu jednotlivých spojení v sieti. Každá stanica periodicky, a taktiež v prípade zmeny stavu, broadcastuje informácie o svojich spojeniach.

Každá stanica si pre každé spojenie v sieti pamätá jeho stav. Pokiaľ neprijme informáciu o zmene stavu spojenia do nastaveného času expirácie, spojenie je považované za nefunkčné. Zlúčením informácií o stave spojení si stanica dokáže zostrojiť graf, v ktorom si udržiava najkratšie cesty ku všetkým ostatným staniciam. Túto informáciu následne používa na smerovanie dát v sieti.

Výhodou algoritmu *LS* je vysoká rýchlosť konvergencie pri zmene topológie siete.

Algoritmus však nie je vhodný pre väčšie siete. Ak je počet pripojených staníc N, počas jedného intervalu obnovy je kvôli broadcastovaniu nutné preniesť každým spojením $O(N^2)$ správ.

Typickým reprezentantom protokolov založených na algoritme *Link State* je *OSPF*[1] [Moy98].

3.3.2 Algoritmus *Distance Vector (DV)*

Každá zúčastnená stanica si udržiava smerovaciu tabuľku – vektor najkratších vzdialeností k cieľovým staniciam. Tento vektor zakaždým po uplynutí obnovovacieho intervalu posiela všetkým susedným staniciam. Po prijatí vektora vzdialeností od susednej stanice zvýši všetky prijaté vzdialenosti o 1 a prepočíta vlastný vektor vzdialeností.

Obnovovací interval zvykne byť implementovaný pre každú cieľovú stanicu samostatne - časovač sa spustí vždy, keď sa susedným staniciam odošle informácia pre danú cieľovú stanicu. Ak sa počas behu časovača táto informácia zmenila, po expirácii časovača sa odošle nová informácia. Ak sa informácia zmenila a časovač nie je spustený, nová informácia je odoslaná okamžite.

Nevýhodou tohto algoritmu je relatívne vysoká doba konvergencie. Čas, ktorý uplynie počas obnovovacieho intervalu môžeme považovať za akúsi etapu. Potom propagácia zmeny cesty môže trvať v najhoršom prípade toľko kôl, aká je dĺžka najkratšej cesty. V prípade odpojenia stanice zo siete však budú vznikať cyklické cesty a vzdialenosti budú rásť do nekonečna.

[1] Open Shortest Path First

Tento problém čiastočne rieši protokol *BGP*[2] [RLH06] používaný v Internete. Spolu s najkratšou cestou pre danú cieľovú stanicu sa posielajú aj identifikátory staníc po ceste, teda cesta sa dá ľahko skontrolovať, či neobsahuje cyklus. Je to však len čiastočné riešenie problému. Konvergencie v prípade odpojenia stanice zo siete môže v najhoršom prípade trvať toľko kôl, aká najdlhšia cesta v sieti existuje. Ak by bolo možné nejakým spôsobom eliminovať používanie obnovovacieho intervalu bez zapríčinenia prenosu potenciálne exponenciálneho počtu správ, vyriešil by sa tým problém dlhej konvergencie.

Veľkou výhodou tohto algoritmu oproti *LS* je nízky počet prenášaných správ.

3.4 Východisko z problémov reaktívnych a proaktívnych protokolov

Problém škálovatelnosti reaktívnych a proaktívnych protokolov viedol vedcov k myšlienke skombinovať reaktívny a proaktívny prístup dohromady. To znamená, že každá stanica by mala poznať určité topologické informácie, na základe ktorých by vedela efektívnym spôsobom vytvoriť cestu ku ktorejkoľvek cieľovej stanici.

Schopnosť orientácie človeka v reálnom svete motivovala k vytvoreniu topologických štruktúr v Ad-Hoc sieťach. Bežný človek má veľmi dobré znalosti svojho okolia, taktiež pozná štvrte mesta v ktorom býva, väčšie mestá v kraji, kraje v krajine, krajiny v regióne, regióny na kontinente, atď. Pokiaľ poznáme adresu, vieme na túto adresu poslať poštu. Pokiaľ však vieme len meno osoby, je veľmi ťažké lokalizovať ju vo svete, aby sme boli schopný skontaktovať ju.

Podobný prístup sa dá aplikovať aj v smerovacích protokoloch v Ad-Hoc sieťach vybudovaním hierarchických štruktúr. Proaktívna zložka takýchto protokolov je udržiavanie hierarchickej štruktúry a schopnosť doručovania dát ak poznáme plnú hierarchickú cieľovú adresu (ďalej hierarchický identifikátor), zatiaľ čo reaktívna časť zabezpečuje lokalizáciu stanice a nájdenie jej hierarchického identifikátora - túto lokalizačnú službu budeme volať distribuovaný lokalizačný server. Vytvoreniu spojenia bude teda zodpovedať nájdenie hierarchického identifikátora cieľovej stanice.

Pred tým, než budeme v ďalšej kapitole prezentovať vlastné riešenie smerovacieho protokolu si ešte ukážeme doterajšie dosiahnuté výsledky v smerovacích protokoloch v Ad-Hoc sieťach založených na hierarchickej štruktúre.

[2]Border Gateway Protocol

3.5 Protokol *Hierarchical State Routing*

Protokol *HSR* [PGHC99] je založený na hierarchickej topológii a algoritme *LS*. Sieť je udržiavaná v hierarchickej topológii, pričom zvolení šéfovia klastrov v danej úrovni postupujú do vyššej úrovne. Prvá úroveň pozostáva z fyzickej topológie siete. Medzi stanicami v ďalších úrovniach sú vytvorené virtuálne spojenia nad spojeniami z predchádzajúcej úrovne.

Stanica si udržiava pomocou algoritmu *LS* smerovaciu tabuľku klastra do ktorého priamo patrí. To znamená, že ak počet staníc v klastri je N a počet úrovní klastrov je M, potom stanica, ktorá je šéfom klastra na úrovni L si potrebuje pamätať informácie len o $O(N \times L) = O(N \times M)$ staniciach na rozdiel od $O(N^M)$ pri algoritme *LS* bez hierarchie.

Aby sme vedeli doručiť dáta danej cieľovej stanici, potrebujeme vedieť jej hierarchický identifikátor. Ten je tvorený sekvenciou identifikátorov staníc na ceste z najvyššej úrovne až ku danej cieľovej stanici. Doručenie správy danej stanici prebieha tak, že správa je postupne doručovaná do vyšších úrovní hierarchie až sa dostane do spoločného klastra zdrojovej a cieľovej stanice. Následne je správa doručovaná dole po hierarchii až kým nie je doručená cieľovej stanici.

Výhoda tohto prístupu je v minimalizovaní veľkosti smerovacej tabuľky a prenášaných smerovacích informácií. Nevýhodou je však potencionálne zahltenie šéfov klastrov, keďže všetky prenášané dáta medzi klastrami sú smerované cez ich šéfov. Ak by bola najvyššia úroveň hierarchie tvorená práve dvoma stanicami, sieť by bola rozdelená na dve veľké klastre a všetky dáta medzi týmito klastrami by boli smerované práve cez tieto dve stanice.

Ďalší problém, s ktorým sa stretávame pri hierarchických sieťach, je zistenie hierarchického identifikátora danej stanice. Protokol *HSR* neumožňuje zistenie hierarchického identifikátora stanice z jej vlastného identifikátora, ale namiesto toho zavádza logické adresovanie. Distribuovaný lokalizačný server teda bude mať za úlohu prevádzať logickú adresu na hierarchický identifikátor. Každej stanici je priradená adresa tvaru $\langle subnet, node \rangle$, kde *subnet* reprezentuje podsieť, v ktorej sa nachádza daná stanica *node* – podobne ako je to v protokole *IP* [Pos81a],[DH98]. Daná podsieť korešponduje s príslušnou skupinou užívateľov a je asociovaná s domácim agentom. Na rozdiel od mobilných extenzií protokolu *IP* môže však byť tento domáci agent v Ad-Hoc sieti taktiež mobilný. Dôležité je poznamenať, že logická sieť je tvorená stanicami, ktoré sa v danom okamihu môžu nachádzať v rôznych častiach fyzickej siete a teda aj v rôznych klastroch.

Predpokladajme, že v každej logickej sieti je jasne určený domáci agent. Hierarchický identifikátor domácich agentov sa bude propagovať v hierarchii smerom nahor, pričom stanice zúčastnené v najvyššej úrovni hierarchie si

navzájom tieto identifikátory povymieňajú. Následne sa tieto identifikátory budú šíriť smerom nadol, teda každá stanica sa dozvie hierarchické identifikátory všetkých logických sietí.

Každý člen logickej siete sa teda dozvie hierarchický identifikátor svojho domáceho agenta a zaregistruje u neho svoju aktuálnu adresu. Registrácia prebieha pravidelne v časových intervaloch a zakaždým, keď nastane dôležitá udalosť, napríklad zmena hierarchického identifikátora domáceho agenta alebo presun stanice do iného klastra. Domáci agent vymaže adresu stanice z databázy, ak nie je obnovený v stanovenom čase.

Keďže každá stanica pozná všetky hierarchické identifikátory domácich agentov, v prípade potreby komunikácie s cieľovou stanicou najskôr kontaktuje jej domáceho agenta, aby zistila hierarchickú adresu cieľovej stanice. Akonáhle túto adresu zistí, ďalšia komunikácia medzi stanicami už môže prebiehať priamo.

Vidíme, že smerovanie komunikácie cez šéfov klastrov neumožňuje veľkú škálovatelnosť protokolu. Taktiež sa vyžaduje, aby centrálna autorita prideľovala okrem unikátnych identifikátorov staníc aj logické adresy.

Kapitola 4

Protokol *Scalable Hierarchical Ad-Hoc Routing – SHAR*

V predchádzajúcej kapitole sme naznačili možné riešenie problémov reaktívnych a proaktívnych protokolov použitím hierarchickej štruktúry. Taktiež sme si prezentovali protokol *HSR*, ktorého hlavný problém spočíval v smerovaní dát cez šéfov klastrov.

Táto kapitola prezentuje protokol *SHAR* - škálovatelný smerovací protokol pre Ad-Hoc siete využívajúci hierarchickú štruktúru.

4.1 Požiadavky protokolu

4.1.1 Identifikátor a hodnosť stanice

Každá stanica S implementujúca protokol $SHAR$ má centrálnou autoritou pridelený unikátny celočíselný identifikátor ID_S z intervalu $\langle 1, ID_{MAX})$. Pri všetkých úvahách budeme ďalej predpokladať, že distribúcia identifikátorov medzi stanice je náhodná. V sieti potrebujeme vytvoriť hierarchickú štruktúru. Parametrom tejto štruktúry bude prirodzené číslo N – očakávaná veľkosť klastra. Celé číslo $RANK_S = max\{r|ID_S \equiv 0 \pmod{N^r}\}$ bude udávať hodnosť stanice – počet úrovní hierarchickej štruktúry, v ktorých je táto stanica šéfom klastra. Keďže $RANK_S$ závisí len na identifikátore stanice, hodnosť stanice, je pevne daná. Teda stanica S je napevno šéfom na úrovniach $0, 1, \ldots, RANK_S - 1$. Všimnime si, že platí $\frac{|\{S|RANK_S=r\}|}{|\{S|RANK_S=r+1\}|} \approx N$, teda očakávaná veľkosť klastra je približne N. Najvyššiu úroveň budeme označovať $RANK_{MAX} = max\{RANK_S\}$. Všetky stanice v sieti patria do jediného klastra na tejto úrovni – je to jediný klaster v sieti, ktorý nemá šéfa. Pre

účely protokolu *SHAR* bude mať tento klaster nevlastného šéfa s vyhradeným identifikátorom 0. Každá stanica bude mať v danom čase práve jeden hierarchický identifikátor HID_S tvorený postupnosťou (ID_S, ID_{M_0}, ID_{M_1}, ..., $ID_{M_{RANK_{MAX}-1}}$), kde stanice M_0, M_1, ..., $M_{RANK_{MAX}-1}$ sú šéfmi nejakých klastrov na úrovniach 0, 1, ..., $RANK_{MAX} - 1$.

4.1.2 Interakcia s nižšou vrstvou

Detekcia dosiahnuteľných staníc a nadväzovanie komunikácie s nimi je mimo rozsah protokolu *SHAR*. Tieto funkcie sú zabezpečované nižšou vrstvou. *SHAR* je notifikovaný o nadviazaní a zrušení spojenia so susednou stanicou asynchrónnymi správami. Cez nadviazané spojenie je možné odoslať dáta – tie sú následne prijaté druhou stranou príjmom asynchrónnej správy.

SHAR predpokladá, že riadiace správy protokolu sú cez nadviazané spojenie prenášané spoľahlivo – to znamená, že správy sú na spojení prijímané v rovnakom poradí ako boli odosielané, bez straty a poškodenia. Odoslanie správy však negarantuje jej doručenie – spojenie môže byť pred tým zrušené. Jeden z protokolov spĺňajúcich túto požiadavku, ktorý môže byť použitý na prenos riadiacich správ, je *TCP*[1][Pos81b].

4.2 Ciele protokolu

Cieľom protokolu je zabezpečiť službu doručovania dát cieľovým staniciam pre vyššiu vrstvu.

4.2.1 Interakcia s vyššou vrstvou

Keďže nad Ad-Hoc sieťami býva častokrát vybudovaná sieť založená na protokole *IP*[2] [Pos81a],[DH98], možno protokol *SHAR* v *ISO/OSI* modeli umiestniť medzi linkovú a sieťovú vrstvu. Pre protokol *IP* sa teda *SHAR* javí ako linková vrstva, v ktorej všetky Ad-Hoc stanice tvoria jednu logickú sieť *IP* protokolu. *IP* adresy tejto siete získame zadefinovaním vhodnej bijekcie z identifikátorov staníc.

Pripomeňme, že protokol *IP* doručuje dáta vrámci logickej siete v dvoch etapách:

[1] Transmission Control Protocol
[2] Internet Protocol

Zistenie linkovej adresy cieľovej stanice – použitím externého protokolu sa zistí linková adresa pre danú *IP* adresu. Zistená adresa môže byť externým protokolom uschovaná po dobu komunikácie s cieľovou stanicou.

Odoslanie dát – linková vrstva vie doručiť dáta so znalosťou linkovej adresy.

Ukazuje sa, že pre linkovú adresu je vhodné použiť hierarchický identifikátor danej stanice. Protokol *SHAR* by mal teda reagovať na požiadavku doručenia dát stanici s daným hierarchickým identifikátorom a požiadavku zistenia hierarchického identifikátora z identifikátora stanice. Obidve požiadavky sú, podobne ako pri interakcii s nižšou vrstvou, implementované pomocou asynchrónnych správ. Opačným smerom *SHAR* posiela asynchrónne správy dvoch typov – doručenie dát lokálnej stanici a odpoveď na požiadavku zistenia hierarchického identifikátora.

4.3 Prehľad protokolu

V predchádzajúcom texte sme identifikovali dve dôležité funkcie protokolu. Toto rozdelenie ešte viac zjemníme a *SHAR* rozdelíme na viacero menších protokolov. Táto sekcia umožní čitateľovi nahliadnuť do celkovej funkčnosti protokolu pred tým, ako prejdeme k jeho formálnej špecifikácii.

4.3.1 Vyhľadávanie najkratších ciest

Protokol *Path Find – SHAR-PF* je pomocným protokolom zabezpečujúcim vyhľadávanie najkratších ciest k dôležitým staniciam.

4.3.2 Vytváranie a udržiavanie klastrov

Aby bolo možné vytvoriť hierarchický identifikátor stanice, je nutné najskôr vybudovať hierarchickú štruktúru klastrov. Túto úlohu bude vykonávať protokol *Cluster Form – SHAR-CF*. Prvých $RANK_S + 1$ členov hierarchického identifikátora stanice S je jednoznačne daných – je to identifikátor stanice ID_S. Jednou z úloh *SHAR-CF* bude nájsť vhodný klaster na úrovni $RANK_S$, ktorého členom sa stanica stane. Jedným z prirodzených kritérií je voľba takého klastra, ktorého šéf je najbližšie. Toto kritérium však nezaručuje súvislosť klastrov a snahy upraviť toto kritérium vedú k nestabilite voľby nadradeného klastra.

Ukazuje sa, že schodnejšie riešenie je akceptácia súčasného členstva stanice vo viacerých klastroch na tej istej úrovni zabezpečujúca ich súvislosť.

Výstupom protokolu *SHAR-CF* je hierarchický identifikátor stanice a informácia o členstvách stanice v klastroch na jednotlivých úrovniach.

4.3.3 Objavovanie členov klastrov

Schopnosť doručiť dáta stanici s daným hierarchickým identifikátorom vyžaduje znalosť o podradených klastroch v ich priamo nadradených klastroch, v ktorých má stanica členstvo. Úlohou protokolu *Cluster Discovery – SHAR-CD* bude distribuovať tieto informácie naučené z *SHAR-CF* vnútri klastrov a udržiavať smerovacie informácie o optimálnych cestách k ich podradeným klastrom.

4.3.4 Doručovanie dát

Systémové požiadavky na doručenie dát stanici s daným hierarchickým identifikátorom bude vybavovať protokol *Data Forward – SHAR-DF*. Protokol najskôr identifikuje najnižší spoločný klaster a dáta prepošle v smere požadovaného podradeného klastra. Tento smer zistí z protokolu *SHAR-CD*.

4.3.5 Distribuovaný lokalizačný server

Stanica sa pri zmene svojho hierarchického identifikátora registruje na distribuovanom lokalizačnom serveri. Hierarchický identifikátor je uschovaný na stanici deterministicky určenej z identifikátora stanice distribuovaným algoritmom.

Systémové požiadavky lokalizácie danej stanice sú spracovávané podobným spôsobom. Taktiež je spustený distribuovaný algoritmus určujúci stanicu disponujúcu s danou informáciou, ktorá následne vráti uložený záznam.

Protokol *Location Server – SHAR-LC* popisuje túto činnosť.

4.4 Formát správ

Táto sekcia formálne opisuje formát asynchrónne prijímaných a posielaných správ v komunikácii so systémom.

Správu chápeme ako postupnosť objektov a zapisujeme ju v tvare $\langle object_1, object_2, \ldots, object_n \rangle$. Všetky správy v komunikácii so systémom majú tvar $\langle type, \ldots \rangle$, kde *type* označuje typ prenášanej správy. Formáty jednotlivých typov správ si opíšeme v ďalšom texte.

4.4.1 Správa *Connection Open*

Vytvorenie nového spojenia nižšou vrstvou je signalizované príjmom tejto správy. Správa má tvar ⟨*connection_open*, *connection*, *node_id*, *distance*⟩, kde *connection* je odkaz na spojenie so stanicou s identifikátorom *node_id* používaný pri ďalšej manipulácii s týmto spojením. Kladné celé číslo *distance* je komunikačná vzdialenosť ku stanici. Táto správa je po príjme preposlaná protokolom *SHAR-CF* a *SHAR-CD*.

4.4.2 Správa *Connection Close*

Táto správa signalizuje zrušenie existujúceho spojenia. Tvar správy ⟨*connection_close*, *connection*⟩ obsahuje odkaz na spojenie *connection*. Správa je po príjme preposlaná protokolom *SHAR-CF* a *SHAR-CD*.

4.4.3 Správa *Connection Receive*

Správa *Connection Receive* informuje *SHAR* o prijatí správy tvaru ⟨*connection_receive*, *connection*, *protocol*, ...⟩ zo spojenia *connection*. Každá takáto správa obsahuje identifikátor *protocol* jedného zo 4 protokolov, pre ktorý je určená. Po prijatí správy je jej časť *protocol* vypustená a preposlaná danému protokolu.

4.4.4 Správa *Connection Send*

Po prijatí požiadavky protokolu *protocol* na odoslanie správy tvaru ⟨*connection_send*, *connection*, ...⟩ doplní *SHAR* do správy informáciu o protokole a správu v tvare ⟨*connection_send*, *connection*, *protocol*, ...⟩ prepošle systému.

4.4.5 Správa *Data Receive*

Túto správu preposiela *SHAR* systému po prijatí z protokolu *SHAR-DF*. Tvar správy je ⟨*data_receive*, *source*, *data*⟩, kde *source* je hierarchický identifikátor odosielateľa a *data* sú posielané dáta.

4.4.6 Správa *Data Send*

Požiadavka vyššej vrstvy na odoslanie dát *data* stanici s hierarchickým identifikátorom *destination* je signalizovaný príjmom správy tvaru ⟨*data_send*, *destination*, *data*⟩. Táto správa je po prijatí preposlaná protokolu *SHAR-DF*.

4.4.7 Správa *Location Query*

Dotaz na vyhľadanie hierarchického identifikátora stanice z jej identifikátora *node_id* je vyššou vrstvou signalizovaný správou tvaru ⟨*location_query*, *query_id*, *node_id*⟩, kde *query_id* je identifikátor požiadavky vrátený v správe *Location Response*. Správa sa po príjme doručená protokolu *SHAR-LS*.

4.4.8 Správa *Location Response*

Odpoveď protokolu *SHAR-LS* na správu *Location Query* tvaru ⟨*location_response*, *query_id*, *node_hid*⟩ je preposlaná systému. Správa obsahuje zistený hierarchický identifikátor stanice *node_hid*, ktorý môže nadobúdať špeciálnu hodnotu *null* v prípade, že sa nepodarilo lokalizovať danú stanicu.

Kapitola 5

Protokol *Path Find* – *SHAR-PF*

SHAR-PF je pomocný protokol, ktorého účelom je hľadanie najkratších ciest k daným staniciam. Protokol je založený na kombinácii algoritmov *Distance Vector* a *Link State*. Jeho činnosť je riadená ostatnými protokolmi *SHARu* a teda nie je možné bez bližších súvislostí priamo popisovať jeho vlastnosti.

5.1 Činnosť protokolu

Dôležitými prvkami smerovacích protokolov je hľadanie najkratších ciest k jednotlivým staniciam v Ad-Hoc sieti. V protokole *SHAR* bude potrebné hľadať najkratšie cesty k význačným staniciam – napríklad stanica sa môže pri výbere klastra, do ktorého patrí, riadiť podľa dĺžky najkratšej cesty k šéfovi klastra. Každá stanica bude mať informácie o niektorých iných staniciach a najkratších cestách k nim. Niektoré z týchto staníc budú natoľko význačné, že bude potrebné susedným staniciam propagovať informáciu o nich. O tom, ktoré stanice sú význačné, budú rozhodovať ďalšie protokoly *SHARu*. Toto rozhodovanie ovplyvní dostupnosť informácií o topológii siete a tým aj optimalitu nájdených ciest.

Každá stanica si bude na každom spojení udržiavať štruktúru orientovaných ciest k množine staníc prijatú od susednej stanice. Tieto informácie skombinuje do štruktúry popisujúcej najkratšie cesty ku všetkým staniciam nachádzajúcich sa v týchto množinách ciest. Tieto cesty sú najkratšie vzhľadom na prijaté informácie zo susedných staníc. Ostatné protokoly môžu informácie o dĺžke týchto najkratších ciest a ich smere používať a určia množinu význačných staníc. Štruktúru popisujúcu množinu najkratších ciest ku všetkým význačným staniciam ďalej stanica odosiela jej susedným staniciam.

Rozumnou štruktúrou popisujúcou najkratšie cesty k množine staníc je strom. Vrcholmi stromu budú stanice, jednotlivé hrany korešpondujú s orientovanými spojeniami medzi príslušnými stanicami. Koreňom tohto stromu je stanica, z ktorej dané cesty vychádzajú. Každý list tohto stromu je význačná stanica. Význačné stanice však nemusia byť nutne listami. Dokážeme si nasledovné tvrdenie o tom, že takýto strom vždy existuje:

Lema 1 *Ak máme daný súvislý orientovaný podgraf*[1] *siete a podmnožinu jeho vrcholov tvoriacu význačné stanice, je možné pre každý vrchol tohto podgrafu zostrojiť strom s koreňom v tomto vrchole obsahujúci všetky význačné stanice, ktorý spĺňa podmienky uvedené vyššie.*

Dôkaz: Strom budeme budovať postupne. Na začiatku obsahuje strom jeden vrchol – koreň. Pre každú význačnú stanicu nájdeme najkratšiu cestu k nej. K stromu potom pridáme najkratší koniec tejto cesty obsahujúci práve jeden vrchol nachádzajúci sa už v strome. Cena zvyšnej časti cesty je určite rovnaká ako cena cesty v strome po tento vrchol. V opačnom prípade by nastal spor s minimálnou dĺžkou cesty. □

Stromové štruktúry prijaté z jednotlivých spojení následne protokol skombinuje do jedného orientovaného grafu. Tento graf bude navyše implicitne obsahovať hrany korešpondujúce s lokálnymi orientovanými spojeniami smerom von. Zo vzniknutého grafu sa vygeneruje strom obsahujúci význačné stanice, ktorý je následne odoslaný všetkým susedným staniciam. Pred odoslaním stromu konkrétnej stanici je však spustený filter, ktorý odsekne zo stromu prípadný výskyt stanice, ktorej je tento strom posielaný.

Niektoré protokoly vyžadujú hľadanie najkratších ciest len vrámci určitých súvislých množín staníc. Každá takáto množina bude mať vlastné označenie. Množina popisujúca všetky stanice bude mať označenie *all*. Stanica bude význačná len vzhľadom na označenie množiny staníc. Takisto počítanie stromu najkratších ciest z grafu a výmena týchto stromov medzi stanicami bude prebiehať pre každé označenie nezávisle. Dá sa to chápať tak, že pre každé označenie existuje nezávislá inštancia protokolu.

5.2 Štruktúra protokolu

Protokol *SHAR-PF*, ktorého činnosť sme si opísali v predchádzajúcej sekcii, je v podstate modifikáciou algoritmu *Distance Vector*. Aby sme však umožnili

[1] Pod podgrafom siete rozumieme graf, ktorého množina vrcholov aj hrán je podmnožinou množiny vrcholov a hrán siete.

rýchlejšiu konvergenciu a obmedzenie počtu správ, pridáme do neho prvky algoritmu *Link State*. Namiesto posielania kompletného stromu pri každej zmene budeme posielať len inkrementálne zmeny jednotlivých spojení.

Každá stanica si bude evidovať databázu spojení – takéto evidované spojenie pomenujeme fragment cesty, alebo skrátene fragment. Každý fragment bude obsahovať nasledovné údaje:

from, to - definuje orientované spojenie medzi stanicami s identifikátormi $from$ a to.

version - definuje poradové číslo verzie ceny tohto spojenia. Čím väčšie je poradové číslo, tým novší je záznam. Poradové číslo je lokálne vzhľadom na dvojicu ($from$, to) a je generované stanicou $from$.

distance - cena prenesenia dát zo stanice $from$ na stanicu to týmto spojením. Špeciálna hodnota ∞ znamená, že spojenie bolo zrušené.

Navyše bude každý fragment obsahovať zoznam spojení, cez ktoré bol tento fragment prijatý. Pri každom spojení v zozname sa bude ešte evidovať zoznam označení množín staníc, do ktorých tento fragment patrí. Každý takýto zoznam bude implicitne obsahovať označenie *all*. Zoznam označení bude existovať aj priamo vo fragmente. Bude obsahovať práve tie označenia, ktoré sa nachádzajú v aspoň jednom zozname označení nejakého spojenia.

Okrem záznamov prijatých cez spojenia budú v databáze uložené záznamy fragmentov obsahujúce informácie o všetkých vychádzajúcich spojeniach. Identifikátor stanice je uložený v položke $from$. Pri každej zmene ceny spojenia sa okrem položky *distance* zmení aj položka $version$ – zvýšením o 1. Zoznam označení množín bude obsahovať zjednotenie všetkých označení množín vo fragmentoch prijatých cez toto spojenie.

Protokol bude od susedných staníc prijímať správy popísané v ďalšej sekcii. Tieto správy budú modifikovať databázu spojení, z ktorej bude pre každé označenie množiny vypočítavaný strom najkratších ciest vedúcich k význačným staniciam daného označenia. Zoznam týchto význačných staníc bude generovaný ostatnými protokolmi. Pre každú susednú stanicu budú tieto stromy najkratších ciest odseknuté v prípadných výskytoch identifikátora príslušnej susednej stanice. Následne budú tieto stromy rozložené do množiny fragmentov, ktorých zmeny budú odosielané príslušnej susednej stanici.

5.3 Formát správ

Jediné správy, ktoré *SHAR-PF* prijíma sú správy *connection_open*, *connection_close* a *connection_receive* definované v špecifikácii protokolu

SHAR. Správa *connection_receive* má tvar ⟨*connection_receive*, *connection*, *type*, *message*⟩, kde *type* je *SHAR-PF*-špecifický typ správy a *message* je správa protokolu *SHAR-PF*. Táto sekcia formálne popisuje formát jednotlivých typov správ protokolu *SHAR-PF* a obsluhu správ *connection_open* a *connection_close*.

5.3.1 Správa *Fragment Update*

Po prijatí tejto správy tvaru ⟨$from$, *to*, *version*, *distance*⟩ sa v databáze fragmentov nájde záznam zodpovedajúci dvojici ($from$, *to*). Pokiaľ takýto záznam neexistuje, je vytvorený a je k nemu priradený ukazovateľ na spojenie, ktorým bola prijatá táto správa. V prípade existencie záznamu sa porovnajú čísla verzií. Ak je prijaté číslo verzie nižšie, je naspäť poslaná správa *Fragment Notify* obsahujúca nové číslo verzie a vzdialenosť. Ak je prijaté číslo verzie vyššie, záznam sa aktualizuje a všetkým staniciam, ktorých ukazovateľ na spojenie je pri zázname uložený sa pošle správa *Fragment Notify*. Nezávisle od čísla verzie je napokon k záznamu doplnený ukazovateľ na spojenie, ktorým bola prijatá táto správa.

5.3.2 Správa *Fragment Withdraw*

Táto správa slúži k notifikácii stanice, že informácie o tomto fragmente už nebudú aktualizované. Správa má tvar ⟨$from$, *to*⟩. V databáze fragmentov sa nájde zodpovedajúci záznam a je z neho zrušený ukazovateľ na spojenie, ktorým bola prijatá táto správa. Pokiaľ pri zázname nezostal žiadny ukazovateľ na spojenie po dlhšiu dobu, je tento záznam z databázy vymazaný. Keďže po každom poslaní správy *Fragment Update* informujúcej o výpadku spojenia je vždy následne poslaná správa *Fragment Withdraw*, nie je vhodné záznamy z databázy vymazávať okamžite. Ináč by v prípade prijatia správy obsahujúcej staré číslo verzie hrozilo jej použitie a následné šírenie.

5.3.3 Správa *Fragment Notify*

Táto správa má rovnaký tvar ako správa *Fragment Update* a slúži na informovanie stanice o starom fragmente. Stanica, ktorá prijme túto správu si vyhľadá zodpovedajúci záznam v databáze. Pokiaľ je verzia záznamu v databáze menšia, záznam je upravený a všetkým staniciam, ktorých ukazovateľ na spojenie je asociovaný s týmto záznamom sa prepošle táto správa.

5.3.4 Správa *Fragment Tag*

Správa *Fragment Tag* tvaru $\langle from, to, tag \rangle$ slúži k informovaniu, že fragment reprezentuje spojenie medzi stanicami, ktoré sú členmi množiny označenej *tag*. K príslušnému zoznamu označení pre dané spojenie v danom fragmente v databáze je pridané označenie *tag*. Následne je prepočítaná tabuľka označení daného fragmentu.

5.3.5 Správa *Fragment Untag*

Po prijatí správy *Fragment Untag* je vykonaná inverzná činnosť k činnosti správy *Fragment Tag* rovnakého tvaru.

5.3.6 Správa *Connection Open*

Vytvorenie nového spojenia je obslúžené vygenerovaním nových stromov najkratších ciest k význačným staniciam. Tieto stromy sú následne rozložené na jednotlivé fragmenty a poslané pomocou horeuvedených správ.

5.3.7 Správa *Connection Close*

Obsluha zrušenia spojenia vymaže z databázy fragmentov všetky výskyty ukazovateľa na toto spojenie.

5.4 Výstupy protokolu

Výstupom protokolu *SHAR-PF* je pre každé označenie množiny strom najkratších ciest ku všetkým význačným staniciam daného označenia zadefinovaných v ostatných protokoloch. Označenie množiny spojení *all* je nadmnožinou všetkých množín spojení a teda je vhodné na hľadanie všeobecných najkratších ciest v sieti k daným význačným staniciam.

Nasledujúca lema ukazuje vlastnosti ciest nájdených protokolom *SHAR-PF*:

Lema 2 *Máme dané označenie súvislej množiny staníc M. Potom pre každé dve stanice A, $B \in M$, kde B je význačná stanica v tomto označení pre všetky stanice v M, protokol* SHAR-PF *zabezpečí v konečnom čase, že stanica A pozná najkratšiu cestu k stanici B prechádzajúcu len stanicami množiny M.*

Dôkaz: Výrok dokážeme pre pevne zvolenú množinu M a stanicu B. Použijeme indukciu podľa dĺžky najkratšej cesty z A do B. Prvý krok indukcie

je triviálny. Dĺžka najkratšej cesty je najmenšia, ak zvolíme stanicu $A = B$. Vtedy A pozná cestu do B – je prázdna.

V ďalšom kroku dokážeme, že ak je dĺžka najkratšej cesty zo stanice A do B rovná $K > 0$ a všetky stanice, ktorých vzdialenosť do B je menšia K poznajú najkratšiu cestu, potom protokol zabezpečí, aby stanica A poznala najkratšiu cestu do B. Nech najkratšia cesta z A do B cez stanice z množiny M má tvar $A \rightarrow C \rightarrow \ldots \rightarrow B$. Keďže stanica $C \in M$ a z indukčného predpokladu pozná najkratšiu cestu do B, jej strom najkratších ciest k význačným staniciam obsahuje túto najkratšiu cestu do B. Táto cesta určite neprechádza stanicou A a preto strom obsahujúci túto cestu bol odoslaný stanici A. V opačnom prípade by najkratšia cesta z A do B bola menšia, teda by sme dostali spor. Keďže stanica A prijala túto cestu a jedna z jej význačných staníc pre dané označenie je stanica B, vygenerovaný strom najkratších ciest musí obsahovať aj cestu k stanici B. Nech má táto cesta tvar $A \rightarrow D \rightarrow \ldots \rightarrow B$. V prípade, že $C = D$, našli sme horeuvedenú najkratšiu cestu. V opačnom prípade sme museli nájsť cestu rovnakej dĺžky, ináč by sme dostali spor s minimálnou dĺžkou cesty cez C. Zostáva nám dokázať, že nájdená cesta prechádza iba stanicami z množiny M. Keďže stanica D poslala stanici A fragmenty s daným označením množiny M, musí mať zadefinovanú aspoň jednu význačnú stanicu v tomto označení a teda patrí do množiny M. Použitím indukčného predpokladu dostaneme, že zvyšok cesty z D do A taktiež patrí do množiny M. □

Ďalšie vlastnosti tohto protokolu si bližšie popíšeme v interakcii s protokolmi *SHAR-CF* a *SHAR-CD*.

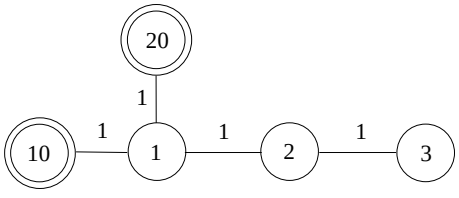
Kapitola 6

Protokol *Cluster Form* – *SHAR-CF*

Táto kapitola obsahuje popis protokolu, ktorého cieľom je vybudovať a udržiavať hierarchickú štruktúru v sieti. Informácie o tejto štruktúre sú využívané ďalšími protokolmi *SHARu*. Dôležitou požiadavkou je udržiavať klastre súvislé. Splnenie tejto požiadavky uľahčí komunikáciu protokolu *SHAR-CD*.

6.1 Činnosť protokolu

Predstavme si, že chceme v sieti vytvoriť klastre na úrovni 0. Každý takýto klaster musí obsahovať práve jedného šéfa – stanicu S, pre ktorú platí $RANK_S > 0$. Teda potrebujeme každú stanicu s hodnosťou 0 priradiť k nejakému klastru. Prirodzeným kritériom pri výbere klastra je dĺžka najkratšej cesty k šéfovi tohto klastra. Splnenie tohto kritéria znamená, že každá stanica si zvolí klaster, ktorého šéf je najbližšie. Ako je vidieť na obrázku 6.1, ak by si stanica 1 zvolila klaster 10 a stanica 2 klaster 20, potom by vznikol nesúvislý klaster 20. Samotné splnenie tohto kritéria teda ešte nemusí zabezpečiť



Obr. 6.1: Sieť s parametrom $N = 10$. Stanice 1, 2 a 3 sú rovnako vzdialené od šéfov úrovne 0 – staníc 10 a 20.

súvislosť vzniknutých klastrov. Tento problém sa dá ľahko napraviť – vždy existuje také priradenie staníc ku klastrom, v ktorom je kritérium splnené a zároveň sú klastre súvislé. Jednoduchý algoritmus, ktorého činnosť je opísaná v nasledujúcom odstavci, vždy nájde požadované priradenie staníc ku klastrom v konečnom čase.

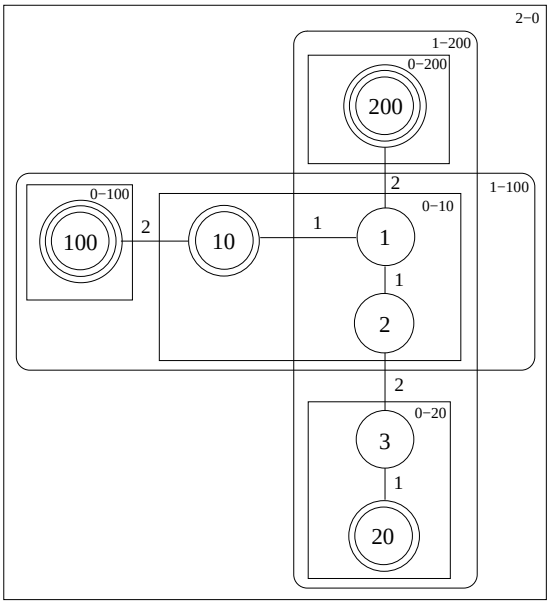
Stanice budú spojeniami prenášať správy formátu $\langle master, distance \rangle$, kde *master* je identifikátor šéfa a *distance* je vzdialenosť k nemu. Každá stanica si bude pre každé spojenie evidovať posledne prijatú takúto správu so vzdialenosťou zväčšenou o dĺžku tohto spojenia. Zakaždým po prijatí správy si stanica z dostupných informácií nanovo zvolí šéfa – vyberie záznam s najnižšou vzdialenosťou. Pokiaľ sa vybraný záznam od posledne zvoleného záznamu líši, stanica tento záznam rozpošle všetkým susedným staniciam. Každý šéf klastra S má pevne zvoleného najbližšieho šéfa $\langle ID_S, 0 \rangle$ – samého seba. Keďže existuje najkratšia cesta z ľubovoľnej stanice k vybranému šéfovi prechádzajúca len cez stanice ktoré si vybrali toho istého šéfa, klastre sú súvislé.

Teraz skúsme v sieti s klastrami na úrovni 0 vybudovať ďalšiu úroveň. Šéf každého klastra na úrovni 0 zvolí svojho šéfa na úrovni 1. Ako ukazuje obrázok 6.2, opäť narážame na problém súvislosti[1] klastrov. Tentokrát však problém nastáva aj v sieťach, v ktorých striktné dodržanie kritéria najmenšej vzdialenosti jednoznačne určuje klastre.

Jedným z možných riešení je umožniť stanici členstvo vo viacerých klastroch tej istej úrovne súčasne. Aby sme upresnili názvoslovie, doteraz bola každá stanica členom nejakého klastra na úrovni 0 a každý klaster na úrovni $L < RANK_{MAX}$ bol členom nejakého klastra na úrovni $L + 1$. Riešením by bolo zaviesť nový typ členstva, kde každá stanica môže byť priamo členom klastra na ľubovoľnej úrovni. Stanica komunikujúca protokolom *SHAR-CF* bude ďalším staniciam propagovať len informácie o klastroch, ktorých je členom. Keďže tieto informácie pochádzajú priamo od šéfov klastrov, znamená to, že z pohľadu tohto nového typu členstva budú klastre súvislé.

Kritérium výberu klastra podľa najbližšieho šéfa vieme rozšíriť pre viacero úrovní. V každej úrovni sa stanica stane členom klastra, ktorého šéf je najbližšie. Aby sme však zachovali hierarchickú štruktúru, je potrebné, aby pre každý klaster ID_1 na úrovni $L < RANK_{MAX}$ existoval taký klaster ID_2 na úrovni $L + 1$, ktorého členmi sú všetky stanice klastra ID_1. Všimnime si, že šéf klastra ID_1 rozhoduje o členstve klastra v klastri na úrovni $L + 1$ hľadaním najbližšieho šéfa s hodnosťou aspoň $L + 2$. Šéf klastra ID_1 sa však

[1]Ku stanici 20, ktorá je šéfom klastra úrovne 0, je najbližší šéf klastra úrovne 1 stanica 200. Jediná cesta spájajúca stanice 20 a 200 však prechádza cez stanice 1 a 2 prislúchajúce klastru 10 na úrovni 0 a teda aj klastru 100 na úrovni 1.



Obr. 6.2: Príklad siete ($N = 10$) obsahujúci prekrývajúce sa klastre.

už stal členom nejakého klastra ID_2. Ak zabezpečíme rozdistribuovanie tejto informácie vrámci členských staníc klastra ID_1, tieto stanice môžu vstúpiť do jeho nadradeného klastra ID_2.

Názorný príklad je možné vidieť na obrázku 6.2, kde stanica 1 je členom klastrov 10 a 200 z dôvodu najmenšej vzdialenosti a zároveň aj členom klastra 100 z dôvodu členstva klastra 10 v 100.

Všimnime si ešte, že šéf klastra na úrovni $L < RANK_{MAX}$ je členom práve jedného klastra na úrovni $L + 1$. To znamená, že hierarchický identifikátor na mieste identifikátora šéfa klastra na úrovni $L + 1$ je jednoznačne určený z miesta identifikátora šéfa klastra na úrovni L. Týmto je však celý hierarchický identifikátor stanice jasne určený.

V príklade na obrázku 6.2 je možné vidieť, že hierarchický identifikátor stanice 1 musí byť 1, 10, 100. Ak by na poslednom jeho mieste bol klaster 200, nebolo by možné podľa neho doručiť dáta z iných staníc klastra 200 ako 1 a 2.

6.2 Štruktúra protokolu

Protokol *SHAR-CF* zabezpečujúci vybudovanie a udržiavanie hierarchie v sieti podľa predchádzajúcej sekcie bude postavený na modifikácii algoritmu

Distance Vector. Namiesto posielania vzdialenosti s každým záznamom však využije protokol *SHAR-PF* z predchádzajúcej kapitoly kombinujúci prvky algoritmov *Distance Vector* a *Link State*.

SHAR-CF bude medzi susednými stanicami vymieňať smerovacie záznamy obsahujúce nasledovné položky:

level - určuje číslo úrovne, v ktorej hľadáme najbližšieho šéfa.

destination - identifikátor šéfa klastra v danej úrovni. Každá takáto stanica bude význačnou stanicou v protokole *SHAR-PF*.

hid - hierarchický identifikátor šéfa *destination*.

Tieto záznamy budú na staniciach uložené v nasledovných smerovacích tabuľkách[2]:

InputRIB - tabuľka obsahujúca smerovacie záznamy prijaté od susedných staníc.

LocalRIB - tabuľka obsahujúca záznamy vybrané procesom rozhodovania. V každej úrovni je vybraný maximálne jeden záznam.

OutputRIB - tabuľka obsahujúca záznamy poslané susedným staniciam.

Každá stanica S s kladnou hodnosťou si na začiatku napevno vloží do tabuľky $LocalRIB$ smerovacie záznamy pre úrovne 0, 1, ..., $RANK_S - 1$. Každý takýto záznam bude obsahovať v položkách *destination* a *hid* vlastný identifikátor a hierarchický identifikátor. Záznamy pre ostatné úrovne si každá stanica určí rozhodovacím procesom na základe obsahu tabuľky $InputRIB$.

Implementácia protokolu *SHAR-CF* zabezpečí, aby všetky stanice, ktorých identifikátor sa nachádza v položke *destination* v nejakom zázname tabuľky $LocalRIB$ boli význačnými stanicami v protokole *SHAR-PF* pre označenia množín staníc *all* a (*shar_cf*, *level*, *destination*), kde *level* a *destination* sú hodnoty z daných záznamov tabuľky $LocalRIB$. Označenia *all* slúži k propagácii všeobecných najkratších ciest k tejto stanici. Druhá množina staníc reprezentuje všetky stanice, ktoré si v danej úrovni vybrali šéfa s daným identifikátorom. Účelom tejto množiny je obmedzenie propagácie správ protokolu *SHAR-CF*, čím sa zamedzí cyklickej propagácii týchto správ. Takáto cyklická propagácia by mohla spôsobiť nesúvislosť klastrov.

[2]V angličtine RIB – Routing Information Base

Tabuľku $OutputRIB$ si stanica bude vypočítavať z tabuľky $LocalRIB$. Pre každé spojenie sa do tabuľky $OutputRIB$ vložia tie smery z tabuľky $LocalRIB$, ktorých cesta v protokole *SHAR-PF* v množine označenej $(shar\_cf, level, destination)$ neobsahuje identifikátor stanice na druhej strane spojenia, čím sa zabráni cyklickému propagovaniu daného klastra. Zmeny záznamov v tabuľke $OutputRIB$ sa automaticky odosielajú príslušnej stanici, ktorá záznam uloží do svojej tabuľky $InputRIB$. Po zmene záznamu v tabuľke $InputRIB$ dôjde k spusteniu rozhodovacieho procesu a vybraniu najlepšieho klastra pre danú úroveň.

Pri nadviazaní spojenia medzi dvomi stanicami dochádza k vygenerovaniu záznamov do tabuľky $OutputRIB$ pre dané spojenie, ktoré efektívne spôsobí vzájomnú výmenu smerovacích záznamov. Naopak, pri zrušení spojenia dochádza k vyčisteniu príslušných záznamov z tabuľky $InputRIB$ a následnému spusteniu rozhodovacieho procesu.

Pred tým, ako budeme pokračovať, dokážeme tvrdenia popisujúce základné vlastnosti klastrov:

Lema 3 *Každá stanica si v súvislej sieti pre každú úroveň $L < RANK_{MAX}$ v konečnom čase zvolí šéfa klastra.*

Dôkaz: Budeme dokazovať pre danú úroveň L. Zo vzťahu $L < RANK_{MAX}$ vieme, že existuje aspoň jedna stanica S, pre ktorú platí $L <= RANK_S$. Táto stanica má pre úroveň L napevno zvoleného šéfa – samú seba. Teda existuje aspoň jedna stanica, ktorá má zvoleného šéfa. Ak predpokladáme, že počet staníc v sieti je konečný, stačí nám dokázať, že počet staníc bez zvoleného šéfa bude klesať. V prípade, že počet staníc bez šéfa je 0, veta je dokázaná. V opačnom prípade musí v sieti existovať dvojica staníc A, B, kde A má a B nemá zvoleného šéfa. Potom zrejme A poslala stanici B protokolom *SHAR-PF* cestu k tomuto šéfovi a protokolom *SHAR-CF* informáciu o zvolenom šéfovi. Teda rozhodovací proces má k dispozícii aspoň jeden platný záznam a v konečnom čase zvolí šéfa. □

Lema 4 *Každý klaster bude v konečnom čase súvislý.*

Dôkaz: Predpokladajme, že existuje nesúvislý klaster na úrovni L. Teda existuje komponent klastra, ktorý neobsahuje šéfa S. Zoberme si množinu najkratších ciest zo všetkých staníc tohto komponentu k šéfovi S z protokolu *SHAR-PF* pre označenie $(shar\_cf, L, S)$. Keďže klaster nie je súvislý, existuje minimálny rez týchto ciest tvorený spojenami, ktoré nespájajú členov klastra. Protokol *SHAR-PF* zrejme poslal správu informujúcu o zrušení týchto

ciest pre označenie ($shar\_cf$, L, S). V konečnom čase budú tieto správy propagované v komponente a vtedy stanice v tomto komponente nebudú mať už dôvod zostať členmi daného klastra. □

Keďže z predchádzajúcej vety vyplýva súvislosť sformovaných klastrov, je možné uvažovať o rôznych funkciách výberu najlepšej cesty použitej v rozhodovacom procese. Z celkového pohľadu protokolu *SHAR* je veľmi dôležitá stabilita klastrov. Preto má zmysel hľadať istú rovnováhu medzi stabilitou zvoleného klastra a vzdialenosťou šéfa tohto klastra. Nájdenie konkrétneho kritéria je však nad rámec špecifikácie tohto protokolu.

6.3 Formát správ

Okrem správ *connection_open*, *connection_close* a *connection_receive* definovaných v špecifikácii protokolu *SHAR* prijíma *SHAR-CF* správu *topology_change* z protokolu *SHAR-PF*. Správa *connection_receive* má tvar ⟨*connection_receive*, *connection*, *type*, *message*⟩, kde *type* je *SHAR-CF*-špecifický typ správy a *message* je správa protokolu *SHAR-CF*. Táto sekcia formálne popisuje formát jednotlivých typov správ protokolu *SHAR-CF* a obsluhu správ *connection_open*, *connection_close* a *topology_change*.

6.3.1 Správa *Route Update*

Správa *Route Update* má tvar ⟨*level*, *destination*, *hid*⟩, kde *level* je úroveň, v ktorej je stanica s identifikátorom *destination* šéfom a jej hierarchický identifikátor je *hid*. Po prijatí tejto správy sa aktualizuje záznam smeru v tabuľke $InputRIB$ pre stanicu, od ktorej bola prijatá táto správa. Následne je spustený rozhodovací proces.

6.3.2 Správa *Route Withdraw*

Táto správa signalizuje zrušenie dosiahnuteľnosti šéfa danej úrovne týmto smerom. Tvar správy je ⟨*level*⟩. Záznam smeru je z tabuľky $InputRIB$ pre danú stanicu vymazaný a je spustený rozhodovací proces.

6.3.3 Správa *Connection Open*

Vytvorenie spojenia je obslúžené zaradením sekcií pre toto spojenie v tabuľkách $InputRIB$ a $OutputRIB$. Sekcia v tabuľke $InputRIB$ je na začiatku prázdna, do sekcie v $OutputRIB$ sa vložia vyhovujúce smery z tabuľky $LocalRIB$.

6.3.4 Správa *Connection Close*

Obsluha zrušenia spojenia zruší príslušné sekcie v tabuľkách $InputRIB$ a $OutputRIB$. Následne je spustený rozhodovací proces.

6.3.5 Správa *Topology Change*

Prijatie tejto správy signalizuje zmenu topológie z protokolu *SHAR-PF*. Jednou zo zmien, ktorá nie je signalizovaná horeuvedenými správami, je zmena ceny spojenia. Preto je potrebné nanovo spustiť rozhodovací proces.

6.4 Rozhodovací proces

Úlohou rozhodovacieho procesu je vybrať pre danú úroveň jeden smer spomedzi smerov naučených zo susedných staníc a následne tento smer nainštalovať do tabuľky $LocalRIB$. Tento proces je spúšťaný po spracovaní všetkých typov správ okrem správy *Connection Open*. Jedným z parametrov, na základe ktorých sa stanica môže rozhodovať o príslušnosti ku klastru, je dĺžka najkratšej cesty k šéfovi klastra. Túto informáciu vieme zistiť z množiny *all* protokolu *SHAR-PF* o každom šéfovi, o ktorom máme záznam v tabuľke $InputRIB$.

Okrem kritéria najkratšej cesty sa ukazuje byť vhodné uvažovať o stabilite dosiahnuteľnosti daného šéfa. Stanica by prideľovala body jednotlivým klastrom podľa vzdialenosti ich šéfa a podľa dĺžky trvania informácie o tomto šéfovi. To by znamenalo, že stanica sa môže rozhodnúť pre zmenu klastra v ľubovoľnom čase. Protokol nevylučuje takéto správanie rozhodovacieho procesu.

Návrhy a porovnávanie jednotlivých kritérií voľby najlepšieho klastra sú mimo rozsah tohto protokolu. Rozhodovací proces však musí spĺňať určité podmienky. Klaster zvolený v tabuľke $LocalRIB$ musí byť platný po každej zmene údajov v tabuľke $InputRIB$. To znamená, že v prípade výpadku spojenia signalizovaného správou *Connection Close* je nutné neplatné záznamy v tabuľke $LocalRIB$ nahradiť inými, prípadne ich vymazať.

V prípade, ak stanica S zmení záznam v jej tabuľke $LocalRIB$ pre úroveň $RANK_S$, jej hierarchický identifikátor je potrebné aktualizovať. Zoberie sa hierarchický identifikátor šéfa zvoleného klastra a začiatočná časť obsahujúca identifikátor stanice a šéfov klastrov pre úrovne menšie ako $RANK_S$ sa nahradí identifikátorom stanice. V prípade, že sa hierarchický identifikátor stanice zmenil, je nutné túto zmenu aplikovať na pevne vložené záznamy v tabuľke $LocalRIB$ pre úrovne $0, \ldots, RANK_S - 1$.

Zmeny spravené v tabuľke $LocalRIB$ je potrebné propagovať do tabuľky $OutputRIB$, ktoré sú následne odoslané susedným staniciam.

6.5 Výstupy protokolu

Protokolu *SHAR-CF* má nasledovné výstupy:

Hierarchický identifikátor stanice - je určený vyššie popísaným postupom rozhodovacieho procesu.

Zoznam klastrov, ktorých členom je táto stanica - je to množina, ktorá pre každú úroveň L obsahuje zoznam klastrov v hierarchickom identifikátore najbližšieho šéfa tejto úrovne začínajúceho sa od klastra na úrovni L. Táto množina navyše obsahuje klaster úrovne $RANK_{MAX}$, ktorý má nevlastného šéfa 0.

Zoznam dvojíc príbuzných klastrov susedných úrovní - obsahuje záznam pre každú susednú dvojicu klastrov do ktorých stanica patrí, vrátane záznamu príslušnosti stanice ku klastru na úrovni 0.

Na príklade z obrázku 6.2 je hierarchický identifikátor stanice 2 (2, 10, 100). Zoznam klastrov, v ktorých je stanica členom obsahuje hodnoty (0, 10), (1, 100), (1, 200) a (2, 0), kde prvá časť hodnoty udáva úroveň klastra a druhá časť hodnoty identifikátor jeho šéfa. Zoznam dvojíc príbuzných klastrov obsahuje hodnoty (0, 10, 2), (1, 100, 10), (2, 0, 100) a (2, 0, 200), kde prvá časť hodnoty udáva úroveň klastra, druhá časť identifikátor šéfa klastra na tejto úrovni a posledná časť identifikátor šéfa klastra na nižšej úrovni, respektíve identifikátor konečnej stanice pre najspodnejšiu úroveň.

Kapitola 7

Protokol *Cluster Discovery – SHAR-CD*

Cieľom tejto kapitoly je navrhnúť protokol vymieňajúci informácie o dosiahnuteľnosti podriadených klastrov a staníc vrámci klastrov, ktorých členom je daná stanica implementujúca tento protokol. Tieto informácie budú využívané ďalšími dvomi protokolmi *SHARu*.

7.1 Činnosť protokolu

Predpokladajme dostupnosť informácií z protokolu *SHAR-CF* o členstve stanice v jednotlivých klastroch. Úlohou protokolu *SHAR-CD* bude pre každý takýto klaster úrovne L vypočítať smerovaciu tabuľku obsahujúcu informáciu pre každý jeho podriadený klaster úrovne $L-1$, prípadne každú členskú stanicu, ak $L = 0$. Ak by sme mali k dispozícii protokol riešiaci problém vypočítavania smerovacej tabuľky pre jeden klaster, je možné všetky správy tohto protokolu rozšíriť o informáciu identifikujúcu klaster, ktorého sa týkajú, a tým dostať požadovaný protokol. Budeme sa teda najskôr snažiť vyriešiť problém smerovania vrámci jedného klastra.

Každý takýto klaster obsahuje členské stanice a podriadené klastre[1]. Zatiaľ, čo neexistuje rozumný limit na počet členských staníc – špeciálny klaster 0 na úrovni $RANK_{MAX}$ obsahuje všetky stanice – existuje odhad počtu podriadených klastrov. Z distribúcie identifikátorov staníc a zo spôsobu voľby nadriadeného klastra šéfom klastra vyplýva, že očakávaný počet podriadených klastrov je N. Každá členská stanica pozná z protokolu *SHAR-CF*

[1] V prípade, že úroveň klastra je 0, namiesto podriadených klastrov sú to podriadené stanice.

informáciu, do ktorých podriadených klastrov patrí. Ako je vidieť na obrázku 6.2, počet podriadených klastrov môže byť rôzny – stanica 1 patrí v klastri 0 na úrovni 2 do dvoch podriadených klastrov úrovne 1 – klastrov 100 a 200. V klastri 200 na úrovni 1 však stanica nepatrí do žiadneho klastra úrovne 0.

Každá členská stanica klastra si bude udržiavať smerovacie informácie o všetkých jeho podriadených klastroch. Členské stanice podriadených klastrov budú zdrojom tejto informácie, ostatné stanice v klastri budú túto informáciu vhodným spôsobom propagovať tak, aby každá stanica v klastri poznala najkratšiu cestu k najbližšej členskej stanici každého podriadeného klastra.

7.2 Štruktúra protokolu

Tento proces sa veľmi podobá na spôsob fungovania protokolu *SHAR-CF*. Preto popíšeme len rozdiely medzi týmito dvomi protokolmi.

Smerovacie záznamy budú obsahovať nasledovné položky:

level, cluster - identifikuje klaster, v ktorom sa daný záznam propaguje. V každom takomto klastri bude jedna inštancia protokolu podobného s protokolom *SHAR-CF*.

subcluster - identifikuje podriadený klaster, ku ktorému hľadáme najkratšiu cestu. V protokole *SHAR-CF* sme hľadali najkratšiu cestu k šéfovi danej úrovne *level*.

destination - identifikátor najbližšej stanice patriacej do daného klastra.

Záznamy budú takisto ukladané do tabuliek $InputRIB$, $LocalRIB$ a $OutputRIB$. Ich sémantika je rovnaká. Zmena je v spôsobe fungovania tabuľky $InputRIB$. V rozhodovacom procese sa budú používať len tie záznamy, ktoré sa týkajú klastrov, ktorých je stanica členom. Namiesto označenia množiny ($shar\_cf$, $level$, $destination$) protokolu *SHAR-PF* budeme používať označenie ($shar\_cd$, $level$, $cluster$, $subcluster$, $destination$). Jeho funkcia bude takisto zabrániť cyklickej propagácie záznamov.

Ďalšia zmena je v tom, akým spôsobom vstupujú do protokolu informácie. Stanica, ktorá je členom klastra *cluster* na úrovni *level* a zároveň aj členom jeho podriadeného klastra *subcluster* vloží do tabuľky $LocalRIB$ záznam, ktorého položka *destination* je identifikátorom tejto stanice. V prípade, že úroveň klastra je 0, namiesto podriadeného klastra uvažujeme podriadenú stanicu.

7.3 Formát správ

Protokol bude prijímať tie isté typy správ ako protokol *SHAR-CF*. Rozdielom bude, že vo všetkých správach budú namiesto parametra *level* trojica parametrov *level*, *cluster* a *subcluster*. Naopak, v správe *RouteUpdate* nebude parameter *hid*. Sémantika správ však zostáva oproti protokolu *SHAR-CF* nezmenená.

7.4 Rozhodovací proces

Úlohou rozhodovacieho procesu bude pre každý podriadený klaster vybrať jeho reprezentanta. Hlavným kritériom bude, aby bol tento reprezentant čo najbližšie. Keďže častejšie zmeny týchto reprezentantov nemajú vplyv na hierarchickú štruktúru, ako to bolo v protokole *SHAR-CF*, je možné ich zmeny propagovať agresívnejšie. Návrh vhodného kritéria rozhodovacieho procesu je mimo rozsah tejto práce. Podobne, ako v protokole *SHAR-CF*, však musí rozhodovací proces spĺňať požiadavku výberu korektného reprezentanta v každom čase.

7.5 Výstupy protokolu

Protokol *SHAR-CD* má nasledovné výstupy:

Zoznam podriadených klastrov - získa informácie o podriadených klastroch pre každý klaster, v ktorom sa stanica nachádza.

Najbližší reprezentanti podriadených klastrov - pre každý podriadený klaster zistí najbližšieho reprezentanta tohto klastra.

Kapitola 8

Protokol *Data Forward* – *SHAR-DF*

Táto kapitola opisuje spôsob, akým sú v sieti smerované dáta určené na doručenie cieľovej stanici s daným hierarchickým identifikátorom.

8.1 Činnosť protokolu

Predpokladajme, že stanica má dostupné údaje z protokolov *SHAR-PF*, *SHAR-CF* a *SHAR-CD*. Táto stanica prijme požiadavku zo systému obsahujúcu dáta a hierarchický identifikátor stanice, ktorej majú byť tieto dáta doručené. Protokol *SHAR-DF* sa bude snažiť tieto dáta postupne doručovať do nižších klastrov, v ktorých sa nachádza cieľová stanica, až budú tieto dáta finálne doručené cieľovej stanici.

Prvý krok stanice bude identifikovať najnižší spoločný klaster s cieľovou stanicou. Vlastné klastre si stanica zistí z protokolu *SHAR-CF*, klastre cieľovej stanice pozná z hierarchického identifikátora tejto cieľovej stanice. Spoločný klaster vždy existuje – všetky stanice sú členom klastra 0 na úrovni $RANK_{MAX}$. Stanica sa teda bude snažiť doručiť dáta do podriadeného klastra tohto spoločného klastra, ktorého identifikátor zistíme z hierarchického identifikátora stanica.

Nasledujúcim krokom bude pomocou protokolu *SHAR-CD* zistiť najbližšieho reprezentanta tohto podriadeného klastra. Ak stanica nemá žiadne informácie o tomto podriadenom klastri, odošle zdrojovej stanici informáciu signalizujúcu zmenu topológie. Zdrojová stanica na to zareaguje zistením nového hierarchického identifikátora cieľovej stanici a môže dáta preposlať znova.

Po zistení najbližšieho reprezentanta podriadeného klastra označíme posielané dáta identifikátorom tohto reprezentanta a odošleme ich v smere najkratšej cesty k tomuto reprezentantovi zistenej cez protokol *SHAR-PF*. Pokiaľ po ceste k tomuto reprezentantovi niektorá stanica nenájde príslušný záznam v protokole *SHAR-PF*, signalizuje to zmenu najbližšieho reprezentanta alebo stromu najkratších ciest k nemu. Takáto stanica určí nanovo spoločný klaster a reprezentanta jeho podriadeného klastra.

8.2 Formát správ

Protokol *SHAR-DF* komunikuje so systémom správami *data_send* a *data_receive*. Navyše prijíma správu *connection_receive* tvaru ⟨*connection_receive*, *connection*, *type*, *message*⟩, v ktorej parameter *type* určuje *SHAR-DF*-špecifický typ správy a *message* obsahuje správu tohto typu. Táto sekcia obsahuje popis jednotlivých typov správ protokolu *SHAR-DF* a taktiež obsluhu správy *data_send*.

8.3 Správa *Deliver*

Príjem tejto správy tvaru ⟨*source*, *destination*, *via*, *data*⟩, signalizuje požiadavku na doručenie dát *data* stanici s hierarchickým identifikátorom *destination*, ktorej odosielateľom je stanica s hierarchickým identifikátorom *source*. Reprezentantom najbližšieho nižšieho klastra je stanica s identifikátorom *via*.

Pokiaľ táto správa nie je prijatá stanicou s identifikátorom *via*, pomocou protokolu *SHAR-PF* je zistený smer najkratšej cesty k stanici *via* a v tomto smere je správa odoslaná nezmenená ďalej. Ak sa smer najkratšej cesty nepodarí zistiť, alebo identifikátor stanice je totožný s parametrom *via*, prebehne postup popísaný v predchádzajúcej sekcii a je zistený nový reprezentant *via*, ktorého smerom je správa odoslaná.

Pokiaľ správa dosiahne stanicu, pre ktorú je určená, je systému odoslaná správa ⟨*data_receive*, *source*, *data*⟩.

V prípade, že nie je možné nájsť podriadený klaster a teda nie je možné správu ďalej doručiť, je odoslaná spätným smerom správa *Deliver*, ktorej parameter *data* obsahuje špeciálnu hodnotu *destination_unreachable*. V prípade, že nie je možné nájsť podriadený klaster počas doručovania takejto správy, je správa zahodená.

8.4 Správa *Data Send*

Správou tvaru $\langle data\_send, destination, data \rangle$ systém signalizuje požiadavku na odoslanie dát cieľovej stanici. Táto požiadavka je spracovávaná podobne ako správa *Deliver*.

Kapitola 9

Protokol *Location Server – SHAR-LS*

Cieľom kapitoly je navrhnúť mechanizmus lokalizácie stanice s daným identifikátorom – teda hľadania jej hierarchického identifikátora.

9.1 Činnosť protokolu

Potrebujeme nájsť spôsob, ako z identifikátora stanice deterministickým spôsobom zistiť jej hierarchický identifikátor. V prípade, že by sme sa chceli vyhnúť uchovávaniu informácie o párovaní identifikátora stanice s jej hierarchickým identifikátorom na iných staniciach, jediným riešením by bolo broadcastovať celú sieť, čo však nie je vhodné riešenie ani pre malé siete.

Preto bude zrejme potrebné párovacie informácie uchovávať aj na iných staniciach. Preformulovaná požiadavka teda bude znieť – potrebujeme nájsť deterministický spôsob, ako z identifikátora stanice A nájsť stanicu B, ktorá pozná identifikátor stanice A. Navyše by sme chceli zabezpečiť, aby sa táto stanica B nemenila často.

Tu sa ukazuje ako vhodné využiť existujúcu štruktúru klastrov – tie boli navrhované tak, aby bola ich štruktúra čo najstabilnejšia. Na začiatok predpokladajme, že hierarchický identifikátor každej stanice A je protokolom *SHAR-LS* zapamätaný na práve jednej stanici B v sieti. Deterministické vyhľadávanie tejto stanice B podľa identifikátora stanice A by mohlo fungovať nasledovne. Začneme klastrom 0 na úrovni $RANK_{MAX}$. V klastri, v ktorom vyhľadávame, zoberieme množinu identifikátorov jeho podriadených klastrov. Potom použitím deterministickej funkcie, ktorej parametrami sú táto množina a identifikátor stanice A, vyberieme identifikátor podriadeného klastra, v ktorom sa bude nachádzať stanica B. Pokračujeme týmto klastrom. Takto

sa postupne dostaneme až do klastra na úrovni 0, v ktorom danou funkciou identifikujeme stanicu B.

Ak vieme takýmto spôsobom deterministicky určiť stanicu B z identifikátora stanice A, nič nebráni stanici A v odoslaní svojho hierarchického identifikátora stanici B. Ostatné stanice potom môžu ľahko lokalizovať stanicu A.

Zostáva sa zamyslieť nad deterministickou funkciou, ktorá zabezpečuje výber podriadeného klastra. Je dôležité, aby drobné zmeny v množine podriadených klastrov spôsobovali len malé zmeny vo výbere podriadeného klastra pre daný identifikátor stanice. Vhodnou sa ukazuje byť funkcia, ktorá vyberie podriadený klaster, ktorého identifikátor je najväčší taký, že je menší alebo rovný identifikátoru hľadanej stanice. Pokiaľ takýto klaster neexistuje, vyberie sa klaster s najväčším identifikátorom. Táto funkcia ovplyvní pri každej zmene množiny len zvolené podriadené klastre pre interval identifikátorov zodpovedajúci dvom susedným identifikátorom podriadených klastrov.

Aby mohol doteraz popísaný protokol fungovať, je dôležitá pravidelná registrácia stanice A na stanici B. Takisto je treba vykonať registráciu zakaždým, keď sa zmení hierarchický identifikátor stanice A. Aby bolo možné zväčšiť interval pravidelnej registrácie, je potrebné implementovať rôzne vylepšenia.

Stanica B disponuje dostatkom informácií na to, aby vedela, či po zmene hierarchickej štruktúry má ešte stále uchovávať záznam pre stanicu A. Jedným z vylepšení by mohlo byť, že po zmene hierarchickej štruktúry spôsobujúcej nepotrebnosť uchovávania záznamu by stanica B preposlala požiadavku na zaregistrovanie stanice A novej stanici. Stále sa však môže stať, že stanica B je nepredvídateľne odpojená zo siete a stanica A sa o tom nedozvie.

Preto je vhodné zabezpečiť redundanciu zapamätaných informácií. V závislosti od voľby parametra N je možné vyššie definovanú funkciu rozšíriť, aby jej výstupom nebol len jeden podriadený klaster, ale viacero. Informácia by bola následne pamätaná vo všetkých podriadených klastroch. Nech je tento počet M. Potom počet staníc pamätajúcich si informáciu o danej stanici je možné zhora ohraničiť číslom $M^{RANK_{MAX}}$. Pri voľbe $N = 16$, $M = 3$ a počte staníc $K = 2^{32}$ by si v priemere každá stanica potrebovala pamätať informácie o $M^{log_N K} = 3^8 = 6561$ staniciach. Vhodným zabezpečením dostatočnej redundancie je možné úplne eliminovať nutnosť pravidelnej registrácie.

Ďalším možným vylepšením pre siete, v ktorých dominuje lokálna komunikácia, je vykonávať registráciu nielen začatím v klastri 0, ale vo všetkých klastroch, do ktorých daná stanica patrí. Algoritmus lokalizácie stanice by sa upravil tak, že vyhľadávanie by začalo v najnižšom klastri. Ak by stanica nebola nájdená, začiatok vyhľadávania by sa posunul o úroveň vyššie.

9.2 Zhodnotenie

Prezentovali sme, akým spôsobom je možné lokalizovať stanice podľa ich identifikátorov. Navrhli sme rôzne vylepšenia, ktoré zlepšujú škálovatelnosť tohto protokolu. Presný popis štruktúry protokolu a formátu vymieňaných správ je však mimo rozsah tejto práce, nakoľko princíp funkčnosti lokalizačného servera *SHAR-LS* je podobný použitým prístupom v ostatných protokoloch, ako napríklad v protokole *Hierarchical State Routing*.

Kapitola 10

Záver

V tejto diplomovej práci sme navrhli a formálne zadefinovali škálovatelný smerovací protokol pre Ad-Hoc siete. Vhodné rozdelenie protokolu *SHAR* na viaceré relatívne nezávislé časti nám umožnilo podrobnejšie skúmať jednotlivé menšie problémy.

Vhodným skombinovaním prístupov algoritmov *Link State* a *Distance Vector* sme vytvorili protokol *SHAR-PF*, ktorého cieľom je efektívne hľadať najkratšie cesty v daných častiach siete. Oddelením šírenia informácií o stave spojení sme v porovnaní s protokolmi založenými na algoritme *Distance Vector* dosiahli výrazné zníženie komunikačnej zložitosti pri zmene topológie siete. Vyhli sme sa však problému periodického broadcastingu stavu liniek známeho z protokolov založených na algoritme *Link State*.

Vybudovaním hierarchickej štruktúry protokolom *SHAR-CF* sme znížili pamäťové požiadavky. Povolením súčasného členstva stanice vo viacerých klastroch na rovnakej úrovni hierarchie sme umožnili lenivú voľbu klastrov, ktorá podstatne znižuje komunikačné nároky stanice pri častých zmenách topológie siete.

Protokolom *SHAR-CD* prenášame informácie o podriadených klastroch, ktorých počet vieme ovplyvniť vhodnou voľbou parametrov protokolu. Tým sme zabezpečili nízke pamäťové nároky na uchovávanie smerovacích informácií.

Obmedzením funkcie šéfov klastrov na tvorbu hierarchie zlepšujeme smerovanie dát, ktoré je zabezpečené protokolom *SHAR-DF*. Na rozdiel od podobných protokolov, dáta nie sú smerované cez šéfov klastrov v hierarchii.

Nakoniec, voľbou vhodného kompromisu medzi spoľahlivosťou, pamäťovými a sieťovými požiadavkami v protokole *SHAR-LS* distribuujeme efektívne lokalizačné informácie o staniciach.

Literatúra

[Bak02] Fred Baker. An outsider's view of manet. 2002. http://w3.antd.nist.gov/wctg/manet/draft-baker-manet-review-01.txt.

[DH98] S. Deering and R. Hinden. Internet protocol, version 6 (ipv6) specification. 1998. http://www.faqs.org/rfcs/rfc2460.html.

[Kah77] R. E. Kahn. The organization of computer resources into a packet radio network. *IEEE Transactions on Communication Technology, Vol. COM-25*, 1977.

[Moy98] J. Moy. Ospf version 2. 1998. http://www.faqs.org/rfcs/rfc2328.html.

[PGHC99] Guangyu Pei, Mario Gerla, Xiaoyan Hong, and Ching-Chuan Chiang. Wireless hierarchical routing protocol with group mobility (WHIRL). Technical Report 990006, 25, 1999. http://www.cs.ucla.edu/NRL/wireless/PAPER/wcnc99.pdf.gz.

[Pos81a] Jon Postel. Internet protocol. 1981. http://www.faqs.org/rfcs/rfc791.html.

[Pos81b] Jon Postel. Transmission control protocol. 1981. http://www.faqs.org/rfcs/rfc793.html.

[RLH06] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4), rfc4271. 2006. http://www.faqs.org/rfcs/rfc4271.html.