

UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Šušmelj

**Implementacija in simulacija
usmerjevalnega protokola RIVER v
NS-2**

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

MENTOR: doc. dr. Mojca Ciglarič

Ljubljana, 2013

Rezultati diplomskega dela so intelektualna lastnina avtorja in Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavlanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje avtorja, Fakultete za računalništvo in informatiko ter mentorja.

Besedilo je oblikovano z urejevalnikom besedil $\text{\LaTeX}$.



Št. naloge: 01921/2013

Datum: 03.04.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANDREJ ŠUŠMELJ**

Naslov: **IMPLEMENTACIJA IN SIMULACIJA USMERJEVALNEGA
PROTOKOLA RIVER V NS-2**
**ROUTING PROTOCOL RIVER: IMPLEMENTATION AND SIMULATION
WITH NS-2**

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Preglejte področje usmerjevalnih protokolov v omrežjih vozil (VANET). Preučite njihove slabosti, ki povzročajo primanjkljaj učinkovitosti, nato pa iz predlogov protokolov, ki jih navaja znanstvena literatura, izberite manjši nabor najbolj obetavnih. Protokol RIVER tudi implementirajte za simulacijo v omrežnem simulatorju NS-2. Izberite metrike, ki najbolje opišejo učinkovitost usmerjevalnega protokola v omrežju VANET. Izbrane protokole nato simulirajte na primernem zemljevidu in z ustrezno gostoto vozil. Protokole primerjajte glede na izbrane metrike in rezultate kritično komentirajte. Nazadnje podajte priporočila, kateri protokoli so realno uporabni v omrežjih vozil in podajte smernice za nadaljnje raziskovalno delo.

Mentor:

doc. dr. Mojca Ciglarič



Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Andrej Šušmelj, z vpisno številko **63050115**, sem avtor diplomskega dela z naslovom:

Implementacija in simulacija usmerjevalnega protokola RIVER v NS-2

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mojce Ciglarič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 9.7.2013

Podpis avtorja:

Zahvala

Zahvaljujem se mentorici dr. Mojci Ciglarič za vodenje in nasvete pri kreiranju diplomskega dela.

Zahvala gre tudi Rašku Pavloviću za tehnično podporo pri implementaciji protokolov v NS-2.

Iskreno bi se zahvalil moji puncu Ani, ki mi je bila ves čas v oporo.

Zahvala gre tudi družini, ki je med študijem veliko pomagala.

Hvala.

Kazalo

Povzetek

Abstract

1	Uvod	1
1.1	Razvoj mobilnih omrežij in omrežij vozil	1
1.2	Lastnosti omrežja vozil	3
1.3	Avtomobilska industrija	4
2	Usmerjevalni protokoli	7
2.1	Topološko usmerjanje	8
2.1.1	Proaktivno usmerjanje	8
2.1.2	FSR	9
2.1.3	Reaktivno usmerjanje	9
2.1.4	DSR	10
2.1.5	TORA	10
2.1.6	AODV	11
2.2	Geografsko usmerjanje	13
2.2.1	Non-DTN	15
2.2.2	GPSR	15
2.2.3	PRB-DV	16
2.2.4	GRANT	17
2.2.5	GPCR	18
2.2.6	GPSR-J+	18

KAZALO

2.2.7	CAR	19
2.2.8	GSR	20
2.2.9	A-STAR	20
2.2.10	STBR	21
2.2.11	GyTAR	21
2.2.12	LOUVRE	22
2.2.13	CBF	22
2.2.14	TO-GO	23
2.2.15	DTN kategorija	23
2.2.16	VADD	24
2.2.17	GeOpps	24
2.2.18	GeoDTN	24
2.2.19	RIVER	25
3	Omrežni Simulatorji	27
3.1	Splošno	27
3.1.1	NetSim	27
3.1.2	OPNET	28
3.1.3	OMNET++	28
3.1.4	NS-2	28
4	Arhitektura NS-2	31
4.1	Dualnost OTcl in C++ okolja	31
4.2	Razredna hierarhija v NS-2	32
4.3	Dogodki v NS-2	33
4.4	Paketi	34
4.4.1	Tipi paketov in glave	34
4.4.2	Metodi <i>send</i> in <i>recv</i>	36
4.5	Vozlišča	36
4.5.1	Ožičeno vozlišče	37
4.5.2	Mobilno vozlišče	38
4.5.3	Konfiguracija in ustvarjanje vozlišč	39

KAZALO

4.5.4	Pošiljanje in sprejemanje paketov	42
4.5.5	Nastavitve v simulacijski skripti	44
4.5.6	Usmerjanje po mobilnem omrežju	46
4.6	Agenti	47
4.6.1	Informacija o stanju agenta	47
4.6.2	Metode in ukazi agentov	48
4.6.3	Kreiranje novega agenta	48
4.7	CMU Monarch	50
4.8	Sledilne datoteke - <i>Trace Files</i>	51
4.8.1	Nastavitve v simulacijski skripti	51
4.8.2	Wireless Trace Format	53
5	Namestitev NS-2	57
5.1	SUMO	57
5.2	Open Street Map	58
5.3	Datoteka <i>Traffic-Connection</i>	59
5.3.1	TCP/IP model v NS-2	60
5.3.2	Primer UDP komunikacije v NS-2	61
5.3.3	Generator prometa CBR	61
6	Implementacija protokola RIVER	63
6.1	OSI model za RIVER	63
6.2	Beacon	64
6.3	Cestni graf	64
6.4	Komponenta nadzora prometa	65
6.4.1	Sonar sporočila	65
6.4.2	Lista povezav - KEL	66
6.5	Potek iskanja poti	66
7	Izvajanje simulacije z NS-2	69
7.1	Glavni gradniki	69
7.2	Simulacijski scenarij	70

KAZALO

7.3	Zagon simulacije	70
8	Analiza dobljenih rezultatov	73
8.1	Simulacijske meritve	73
8.2	AWK skripta	74
8.3	Rezultati primerjav	75
8.3.1	Razmerje med poslanimi in prejetimi paketi	75
8.3.2	Število nepotrebno generiranih poti za usmerjanje	76
8.3.3	Količina poslanih podatkov na sekundo	77
8.3.4	Število posredovanj podatkovnih paketov	78
8.3.5	Časovne izgube	79
8.3.6	Izguba podatkovnih paketov	80
9	Zaključek	81

Seznam uporabljenih kratic in simbolov

NS-2 - Network Simulator 2

WLAN - Wireless Local Area Network

IEEE - Institute of Electrical and Electronics Engineers

TTL - Time To Live

NFP - Nearest with Forwarding Progress

MANET - Mobile Ad Hoc Network

UDP - User Datagram Protocol

TCP - Transmission Control Protocol

IP - Internet Protocol

ARP - Address Resolution Protocol

LL - Link Layer

IFQ - InterFace Queue

MAC - Media Access Control

PHY - PHysical Layer

FTP - File Transfer Protocol

Povzetek

Vozila opremljena z brezžičnimi komunikacijskimi napravami lahko tvorijo omrežje posebne vrste. Ni pomembna samo udobna in interaktivna vožnja, temveč tudi varnost. Avtomobilska industrija se vedno bolj izpopolnjuje na tem področju. Vozila želi razviti do take zmožnosti zaznavanja in komuniciranja, kakor komunicirajo ljudje med sabo. Za usmerjanje podatkov po omrežju obstaja veliko algoritmov. Ustreznost teh algoritmov je v praksi težko ugotoviti. Zato obstajajo omrežni simulatorji, ki so nam v pomoč pri testiranjih. Diplomsko delo zajema predstavitev omrežnega protokola RIVER, ki spada med novejšje iz področja omrežja vozil. S svojo komponento nadzora prometa ter implicitnim pošiljanjem sporočil po omrežju, izpopolnjuje nekatere pomanjkljivosti že obstoječih protokolov. V diplomskem delu so opisani algoritmi in protokoli, ki so bili podlaga za razvoj marsikaterega usmerjevalnega protokola. Predstavljena so nekatera orodja za simulacijo omrežnih protokolov, kjer je podrobneje opisana arhitektura omrežnega simulatorja NS-2. Predstavljena je implementacija usmerjevalnega protokola RIVER za omrežni simulator NS-2. Simulator omogoča tudi sledenje pošiljanju podatkov po omrežju. Iz sledilnih datotek lahko s pomočjo skriptnih programskih jezikov izločimo pomembne podatke, ki jih lahko uporabimo za meritve. V diplomskem delu je izbranih nekaj metrik za primerjavo protokolov. S skriptami AWK je narejena primerjava metrik med protokolom RIVER ter ožjim naborom usmerjevalnih algoritmov.

Ključne besede:

Omrežja vozil, VANET, RIVER, omrežni simulator NS-2

Abstract

Vehicles equipped with wireless communication devices can form a specific type of network. Not only comfortable and interactive ride is important, but also safety. The automotive industry is increasingly improving in this area. Vehicles want to develop to such perception and communication, as people communicate with each other. There are many algorithms for routing data over a network. The suitability of these algorithms is difficult to determine in practice. Therefore, there are network simulators, which are helpful with testings. Diploma thesis includes a presentation of the RIVER network protocol, which is one of the newer from the network vehicles area. Its traffic component and implicit sending messages across the network, improves some weaknesses of the existing protocols. This diploma thesis describes the algorithms and protocols that were the basis for the development of many routing protocols. Some VANET simulation tools are described and in details is presented architecture of NS-2 network simulator. Implementation of RIVER routing algorithm is also presented for the NS-2 network simulator. NS-2 allows the tracking of sending data over the network. From the trace file, you can with help of scripting programming languages, eliminate important data that can be used for the measurement. This diploma thesis uses the AWK scripts for comparing metrics between RIVER protocol and narrow selection of routing algorithms.

Key words:

Vehicular networks, VANET, RIVER, NS-2 network simulator

Poglavje 1

Uvod

V diplomskem delu bomo pregledali področje omrežja vozil in predstavili orodja, ki so nam v pomoč pri simulaciji usmerjevalnih protokolov. Opisali bomo nekaj zanimivih usmerjevalnih protokolov in podali njihove prednosti in slabosti. Naredili bomo ožji izbor usmerjevalnih protokolov, ter jih s pomočjo omrežnega simulatorja med seboj primerjali. Delo bomo razdelil na tri širša področja. V prvem delu bomo naredili kratek pregled mobilnih omrežij na splošno, ter opisali nekaj predstavnikov usmerjevalnih protokolov, ki se uporabljajo v omrežju vozil. Osrednji del diplomskega dela bo predstavljal podrobni opis delovanja usmerjevalnega protokola RIVER in arhitekture omrežnega simulatorja NS-2. V sklepnem delu pa bomo opisali delo s simulatorjem NS-2, ter nastavitvami za delovanje in izvajanje simulacij. Sklepni del bo vključeval izbiro metrik, s pomočjo katerih bomo lahko nazorno določili, kateri usmerjevalni protokol je boljši.

1.1 Razvoj mobilnih omrežij in omrežij vozil

Začetki mobilnih omrežij segajo v leto 1968, ko so začeli s projektom ALOHA. Omrežje ALOHA je bilo namenjeno povezavi izobraževalnih ustanov na otočju Havaii. Protokol ALOHA [1] je bil t. i. single-hop protokol in je tako imel fiksne bazne postaje, preko katerih je potekala komunikacija med vozlišči

omrežja. Dva vozlišča sta morala tako biti v doletu vseh sodelujočih vozlišč.

ALOHA je bila za tiste čase velik korak v razvoju računalniških omrežij glede na razpoložljivo razpoložljivo znanje in tehnologijo. Po zgledu protokola ALOHA se je leta 1973 razvil protokol, ki je temeljil na multihop omrežju, DARPA. To paketno radijsko omrežje (PRnet) je uporabljalo vozlišča ne samo za direktno komunikacijo med posameznimi vozlišči, temveč tudi za prenos podatkov do ciljnih vozlišč, ki trenutno niso bili v doletu. To pa je glavna razlika med single-hop ter multihop omrežjem.

Vse brezžične enote so uporabljale protokole, ki nikoli niso prišli v komercialno rabo, zato pa se je kasneje uveljavil standard IEEE 802.11 za brezžično omrežje WLAN. S to uvedbo je inštitut IEEE začel uporabljati termin *ad hoc* omrežja.

Celična omrežja so vrsta mobilnih omrežij, kjer je omrežje sestavljeno iz več področij imenovanih celice, ter eno ali več baznih postaj. Mobilno vozlišče (mobilni aparat, pozivnik, različni radijski sistemi itd.), ki je del neke celice, tako komunicira z drugimi vozlišči preko baznih postaj. Celična omrežja izkoriščajo tudi opremljenost vozil z brezžičnimi komunikacijskimi napravami. Te predstavljajo omrežja vozil, ki jih bomo predstavili kasneje.

Poznamo pa tudi računalniška omrežja, kjer se mobilne naprave lahko prosto gibajo, med seboj so pa povezane brezžično. To so t. i. mobilna *ad hoc* omrežja (MANET). Omrežja vozil so tak primer računalniških omrežij, kjer so vozlišča vozila. Vozila se lahko prosto premikajo in komunicirajo z ostalimi vozili. Komunikacija je omogočena s pomočjo brezžičnih komunikacijskih enot, ki so vgrajene v vsako vozilo. Vsako vozilo pa nima naloge samo komuniciranja z drugimi vozili, temveč ima nalogo usmerjevalnika, kjer igra vlogo posrednika pri pošiljanju podatkov po omrežju.

VANET (angl. *Vehicular Ad Hoc Networks*) je tehnologija, ki uporablja vozila kot vozlišča in z njimi tvori mobilno omrežje. Vsako prevozno sredstvo tako sodeluje kot brezžični usmerjevalnik ali vozlišče, ter se povezuje z ostalimi vozlišči v razdalji nekaj 100 metrov, odvisno od razpoložljive tehnolo-

logije. Tako lahko množica vozil tvori zelo široko omrežje. Ker pa je danes zelo pomembna neprekinjena povezava z Internetom, stalna obveščenost o stanju na cestah, itd. se je avtomobilska industrija začela zelo razvijati na tem področju omrežja vozil.

1.2 Lastnosti omrežja vozil

Dinamičnost omrežja: Omrežje uporablja različne napredne protokole za usmerjanje paketov po omrežju. Hitrost in izbira poti sta zelo pomembna dejavnika pri dinamičnosti topologije omrežja. Za primer lahko vzamemo dve vozili, ki se oddaljujejo ena od druge s hitrostjo 100 km/h z razdaljo oddajanja (TR - Transmission range) 250 m . Ti dve vozili bosta tako lahko povezani samo 4.5 s ($250 \text{ m} / 55.6 \text{ m/s}$), kar daje samemu omrežju visoko dinamično topologijo.

Omrežje je večino časa nepovezano: Z zgornjim primerom dinamičnosti omrežja smo torej pokazali, da se mora vozilo v čim krajšem času ponovno priključiti omrežju, če želi biti stalno povezano. Vendar se to zaradi manjšega števila vozil na določenem območju pogosto dogaja. Ta problem izpada povezljivosti omrežja je prisoten predvsem pri vozliščih, ki imajo vlogo usmerjevalca paketov. Namesto da bi posredovali paket naprej po omrežju, izgubijo povezavo z ostalimi vozlišči in tako pride do izpada.

Mobilno modeliranje in predvidevanje: Če želimo imeti konstantno povezljivost omrežja, potem je zelo važno, da znamo predvidevati premikanje samih vozlišč (v našem primeru vozil). Zato si pomagamo z geografskimi kartami ter s pomočjo vedenja o obnašanju vozil (kdaj in kje so gneče v mestih, kakšna je hitrost vozil na določenem odseku poti itd.) lahko učinkovito načrtujemo omrežje vozil.

Raznolikost komunikacijskega okolja: Mobilno omrežje je lahko zelo kompleksno (mesta), ali pa zelo preprosto (avtocesta). Pri avtocestnem

modelu je omrežje zgrajeno zelo preprosto, tudi smeri in hitrosti vozil so takoj jasne. Medtem ko je model mesta bolj zapleteno omrežje. Izbor usmerjevalnega algoritma in kompleksnost omrežja je odvisna od struktur cest, stavb, dreves in podobnih ovir, ki preprečujejo preprosto komunikacijo med vozili.

Omejitev glede količine poslanih podatkov: Ob nesrečah ali hitrem zaviranju lahko vozila med seboj komunicirajo, da en drugega obvestijo o dogodku. Ker pa je zelo važna hitrost dostave paketov, je aplikacija VANET omejena z količino in časom dostave podatkov.

Povezanost z zunanjimi senzorji: Veliko aplikacij VANET uporablja informacije o lokaciji vozila, saj si tako olajša delo pri modeliranju mobilnega omrežja ter pri implementaciji samega usmerjevalnega algoritma.

1.3 Avtomobilska industrija

VANET ali omrežja vozil so v zadnjem desetletju pritegnila veliko pozornost raziskovalnih ustanov in industrije. Sama ideja o tem, da lahko vozila komunicirajo med sabo, je spodbudila raziskovalne ustanove, da raziščejo to področje.

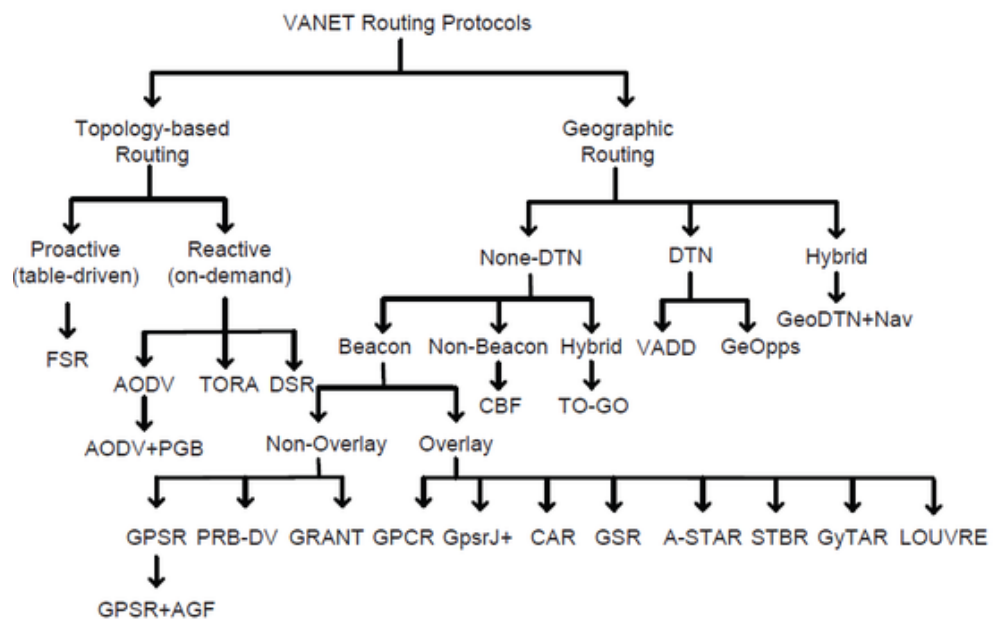
Zaradi visoko zmogljive tehnologije in veliko raziskav na področju brezžične komunikacije in elektronike je napredek v avtomobilski industriji velik. Na cestah se pojavlja vedno več vozil in tako postaja vožnja vedno bolj zahtevna in nevarna. Varnostna razdalja, primerna hitrost in ostali cestno prometni predpisi so vedno manj spoštovani. Zato so se proizvajalci avtomobilov odločili, da združijo moči z raziskovalnimi inštituti in ustanovijo združenja, ki bodo pomagala reševati pomembne situacije na prometnih cestah. Pri korak na tem področju je bila uvedba standarda WAVE (Wireless Access for Vehicular Environment), ki temelji na brezžični komunikaciji med vozili in brezžični komunikaciji med vozili in baznimi obcestnimi postajami. Sistem WAVE temelji na standardu IEEE 802.11p [4], ki se ga uporablja od

leta 2009 naprej. WAVE tehnologija je rešitev za varnost vozil, kjer so vozniki zgodaj obveščeni o nevarnostih na cestah, asistencah, itd. Vozila tako poročajo o stanju na cestah. To poročajo raznim prometnim agencijam, katere potem posredujejo naprej središčem za obveščanje voznikov na širšem območju. Ali pa kot je navedel gospod Xiang, ki dela raziskave na področju tehnologije WAVE, da *WAVE tehnologija omogoča razširitev naravnega zaznavanja voznika in daje možnost zaznave vozilom*. [5]

Poleg tehnologije WAVE obstajajo razna avtomobilska združenja kot so C2CCC (Car 2 Car Communication Consortium), CALM [2], VII [3] in podobni. Obstajajo tudi razni projekti na tem področju, kot so FleetNet, CarNet, NOW - Network On Wheels, CarTALK 2000, VICS - Vehicle Information and Communication System.

Poglavje 2

Usmerjevalni protokoli za omrežja vozil



Slika 2.1: Usmerjevalni protokoli za omrežja VANET [6]

Za navadna mobilna omrežja, kot je celično omrežje, je bilo razvitih že veliko usmerjevalnih algoritmov. Omrežja vozil pa predstavljajo prav poseben

izziv na področju implementacije usmerjevalnih algoritmov. To komunikacijsko omrežje je porazdeljeno in samo-organizirano, sestavljeno iz premikajočih se vozil. Sam mobilni vzorec, ki bi nam pomagal pri načrtovanju algoritma, je težko ugotoviti. Predvidevanje o smeri in vožnji vozil je tako velik dejavnik pri dobri izbiri protokola za usmerjanje.

Slika 2.1 prikazuje dve kategoriji usmerjevalnih protokolov v omrežjih VANET. Topološko usmerjanje (angl. *Topology-based routing*) po omrežju uporablja informacije o trenutnem stanju povezav na omrežju, da lahko pošlje podatke do zelenih ciljev (vozlišč). Geografsko usmerjanje (angl. *Geographic routing*) pa uporablja informacijo o sosednih vozliščih, da izvede nadaljnje posredovanje paketov. Pri posredovanju paketov po omrežju vozil imajo več težav algoritmi za usmerjanje, ki spadajo pod kategorijo Topološkega usmerjanja, saj se povezave na omrežju vozil konstantno spreminjajo.

2.1 Topološko usmerjanje

Kot smo že prej omenili je za topološko usmerjanje značilno, da se pošiljanje paketov po omrežju vozil, izvede na podlagi trenutnega stanja povezav med vozlišči. Naprej se topološko usmerjanje deli na proaktivno in reaktivno usmerjanje.

2.1.1 Proaktivno usmerjanje

Pri proaktivnem (angl. *Proactive*) usmerjevalnem protokolu, je nekje v ozadju vzdrževan podatek o tem, kateremu vozlišču bodo trenutno posredovani podatki (paketi). Ta informacija se obdrži ne glede na trenutno komunikacijo ali zahteve po komunikaciji v omrežju. Kontrolni paketi, ki skrbijo za stanja povezav med dvema vozliščema, se razpošiljajo in poplavljaajo omrežje (angl. *Broadcasting and Flooding*). S tem je ohranjeno tako stanje povezav, kot tudi pot do ciljnega vozlišča. Tako je znotraj vsakega vozlišča ustvarjena tabela, ki hrani podatke o povezavi z sosednjim vozliščem, ki vodi do ciljnega vozlišča. Običajno se zgodi, da je znotraj tabele veliko podatkov o poveza-

vah med vozlišči, kateri nikoli niso uporabljeni. Zaradi tabele se proaktivni algoritem poimenuje tudi *table-driven*.

Slaba lastnost proaktivnega usmerjanja je ta, da zaradi nepotrebnih podatkov v tabelah in konstantnega skladiščenja tabele v vozliščih, pripelje do velike zasičenosti omrežja, zasedenosti razpoložljive pasovne širine in nepotrebnega hranjenja poti do ciljnega vozlišča. Dobre lastnosti pa se izkažejo pri aplikacijah v realnem času (angl. *Real-Time Application*), kjer dostopni čas do podatkov igra pomembno vlogo. Proaktivni usmerjevalni algoritem tako omogoča stalen dostop do tabele, kjer hrani pot do ciljnega vozlišča. Nepotrebnega iskanja novih poti ni.

2.1.2 FSR

(Fisheye State Routing) [7] je najučinkovitejši predstavnik topološkega načina usmerjanja podatkov po omrežju. FSR hrani topološki zemljevid v vsakem vozlišču, sproti pa posodablja tabele samo v sosednjih vozliščih. Tako mu ni potrebno popravljati stanja o povezavah na celotnem omrežju. Stanje povezave na trenutnem vozlišču pošilja trenutnemu ciljnemu vozlišču z različnimi frekvencami. Bližjim vozliščem oddaja sporočila z visokimi frekvencami, bolj oddaljenim vozliščem oddaja z nizkimi frekvencami.

Nastanejo pa tudi pomanjkljivosti pri pošiljanju z različnimi frekvencami. Pri pošiljanju podatkov z nižjimi frekvencami pride do nenatančnega sprejema na strani ciljnega vozlišča. Te pomanjkljivosti se odpravi, ko se paketi bližajo ciljnemu vozlišču.

V majhnih mobilnih omrežjih pa ima ta protokol slabe zmogljivosti. Veliko ima neznanja o oddaljenih vozliščih, ter tako težko ohranja povezanost s ciljnim vozliščem.

2.1.3 Reaktivno usmerjanje

Najznačilnejše za reaktivni usmerjevalni protokol (angl. *Reactive On Demand Routing*) je to, da odpre komunikacijski kanal in s tem pot do vozlišča

samo takrat, ko je zahtevana komunikacija z določenim vozliščem. Protokol ohranja samo tisto pot, ki je trenutno v uporabi. Omrežje je obremenjeno samo takrat, ko je protokol v fazi odkrivanja poti do ciljnega vozlišča. Ko poizvedovalni paketi najdejo želeno pot do končnega vozlišča ali množice vozlišč, se poplavljanje paketov in s tem obremenitev omrežja zaključi.

Slabe lastnosti te vrste protokolov so predvsem dolgotrajno iskanje poti ter poplavljanje s poizvedovalnimi paketi, ki lahko povzroči ne-povezljivost med določenimi vozlišči.

2.1.4 DSR

(Dynamic Source Routing) [8] uporablja tako imenovano *izvorno usmerjanje* (angl. *source routing*). Za izvorno usmerjanje je značilno, da za vzdrževanje poti poskrbi izvorno vozlišče in ne ciljno vozlišče. DSR pošlje poizvedovalne pakete po omrežju. Na poti do ciljnega vozlišča se v poizvedovalne pakete hranijo vsi naslovi vozlišč, ki so posredovala pakete naprej po omrežju. Po prejemu poizvedovalnega paketa, ciljno vozlišče pošlje nazaj podatke o celotni poti do izvora do cilja. Protokol se lahko izboljša tako, da ciljno vozlišče shrani vse prejete poizvedovalne pakete in nazaj do izvirnega vozlišča pošlje seznam več možnih poti do ciljnega vozlišča. Ta izboljšava se dobro izkaže v primeru, ko kakšno mobilno vozlišče, ki je del poti, izstopi iz zaključenega omrežja. DSR ne pošilja poizvedovalnih paketov v intervalih in tako ne zaseda veliko pasovne širine omrežja. Slabosti pa ima predvsem v pošiljanju nepotrebnih paketov po omrežju in s tem obremenjuje omrežje. V visoko mobilnih omrežjih se slabo izkaže, prekinjene povezave pa lokalno ne zna reševati.

2.1.5 TORA

(Temporally Ordered Routing) [9] za razliko od ostalih algoritmov tega tipa, ne uporablja način iskanje najkrajše poti po grafu, ampak uporablja drugačen način. TORA zgradi in vzdržuje usmerjeni aciklični graf (angl. *Directed*

Acyclic Graph), ki ima za koren grafa ciljno vozlišče. To naredi s pomočjo poizvedovalnih paketov. Graf usmerja potek paketov po omrežju in skrbi za dosegljivost vseh vozlišč. Paketi tako potekajo od vozlišč z višjo višino proti vozliščem z nižjo višino. TORA nastopa v treh fazah. Prva faza poskrbi za vzpostavitev poti, DAG grafa, ki ima za korensko vozlišče ciljno vozlišče. Druga faza je vzdrževanje poti, ki mora biti prisotna zaradi mobilnosti in se tako določene povezave prekinejo. Zadnja faza pa nastopi, ko se ugotovi, da določena pot ne vodi do ciljnega vozlišča, in se tako te povezave odstrani.

Prednosti protokola TORA so v tem, da zgradi graf takrat, ko ga potrebuje. Zelo dobro deluje v omrežju z gosto poseljenimi vozlišči. Kljub temu je bolj malo v uporabi, ker obstajajo drugi protokoli, ki se bolje odrežejo (DSR je že eden takih protokolov).

2.1.6 AODV

(Ad Hoc On Demand Distance Vector) [10] je tudi reaktivni protokol in vzpostavlja pot do ciljnega vozlišča samo na zahtevo. Ko neko vozlišče potrebuje povezavo, le ta pošlje zahtevek za povezavo. Poslan zahtevek potuje po omrežju (ostala vozlišča ga posredujejo do ciljnega vozlišča) in si beleži seznam vozlišč, ki jih je obiskal na poti. Ciljno vozlišče tako dobi več možnih poti do izvirnega vozlišča. Ko ciljno vozlišče prejme več kot eno pot, sporoči nazaj izvoznemu vozlišču, potrdilo o vzpostavljeni poti (in s tem seznam vozlišč za pot). Izvorno vozlišče se tako odloči za pot, ki vsebuje manj vozlišč. Neuporabljene poti se kasneje brišejo iz seznama razpoložljivih poti. Če se povezava prekine, je to sporočeno ciljnemu vozlišču.

Protokol AODV poskuša zmanjšati število pošiljanj sporočil po omrežju. Prvo pošiljanje po omrežju se pojavi, ko izvorno vozlišče pošlje zahtevo (sporočilo, ki je zahtevek za povezavo) za pošiljanje podatkovnih paketov do ciljnega vozlišča ali ciljnih vozlišč. Pri AODV paket vsebuje nekaj podatkov, ki jih ostali protokoli na zahtevo nimajo. Sam paket za zahtevo poti (angl. *RouteRequest packet*) vsebuje naslov izvirnega vozlišča (SrcID (angl. source identifier)), naslov ciljnega vozlišča (DestID (angl. destination iden-

tifier)), *source sequence number* (SrcSeqNum), *destination sequence number* (DestSeqNum), *broadcast identifier* (BcastID) in podatek o času aktivnosti paketa (TTL). Glavna razlika med AODV in ostalimi protokoli na zahtevo je podatek *DestSeqNum*. Vozlišče posodobi svojo informacijo o poti (poti od izvora do ciljnega vozlišča) samo takrat, ko je številka *DestSeqNum* večja ali enaka zadnji shranjeni številki *DestSeqNum*. Torej *DestSeqNum* označuje, kako sveža je pot, ki so jo sprejelo ciljno vozlišče. Vozlišče, ki je med izvirnim in ciljnim, najprej prejme paket za zahtevo poti (RouteRequest). Ta paket posreduje naprej ali pa v primeru, da ima podatke o najnovejši poti do ciljnega vozlišča, pošlje paket *RouteReply* nazaj do ciljnega vozlišča. Par *BcastID*-*SrcID* prideta prav, ko vozlišče večkrat prejme *RouteRequest* in se tako večkratne pojavitve brišejo. Vsa vozlišča, ki imajo veljavne informacije o poti do ciljnega vozlišča, lahko pošljejo paket *RouteReply*. Vsako vmesno vozlišče (vozlišče med izvirnim in ciljnim vozliščem) med posredovanjem *RouteRequest* paketa shrani naslov prejšnjega vozlišča in njegov *BcastID*. *TTL* poskrbi, da se ta vnos briše, v primeru da se časovnik izteče pred prejemom podatka *RouteReply*. To pomaga pri vzpostavljanju aktivne poti do ciljnega vozlišča. Ko določeno vozlišče prejme paket s podatkom *RouteReply*, je informacija o naslovu prejšnjega vozlišča, kjer je bil paket prejet, tudi shranjena. Shranjeni naslovi vozlišč, preko katerih je prišla informacija *RouteReply*, pomagajo pri posredovanju podatkovnih paketov do ciljnega vozlišča.

Največja prednost protokola na zahtevo je v tem, da ima s pomočjo podatka *DestSeqNum* informacije o najnovejši poti do ciljnega vozlišča. Čas za vzpostavitev poti je krajša, ker paketom z zahtevo vzpostavitve poti ni treba potovati do ciljnega vozlišča in lahko že prej ugotovijo pot do ciljnega vozlišča. Ima pa tudi slabosti, kot so periodično pošiljanje paketov za vzpostavljanje poti in s tem obremenitev omrežja. Lahko pride tudi do neskladnosti poti, če ima določeno vmesno vozlišče napačne podatke o najnovejši poti do ciljnega vozlišča, ta pot pa sploh ne obstaja. Tudi paketi *RouteReply* lahko zasedejo precejšen del pasovne širine omrežja.

Zaradi zgoraj opisanih slabosti, je Naumov uvedel boljši pristop pri AODV.

Uvedel je mehanizem *Preferred Group Broadcasting*, ki zmanjša pomanjkljivosti AODV, kot so zasedenost omrežja s paketi za zahtevo vzpostavitev poti ter da ne prihaja do poti, ki sploh ne obstajajo. Ta mehanizem prispeva ogromno prednost v omrežjih vozil, kjer se vozila, opremljena z brezžičnimi oddajniki, gibljejo zelo hitro. Pri PGB se gleda prejeti signal. Prejemniki oddanega signala lahko določijo, ali so v skupini, ki vodi do ciljnega vozlišča in komu v skupini se bo posredoval paket z zahtevo po vzpostavitvi poti do ciljnega vozlišča. Ker pa samo eno vozlišče znotraj skupine sodeluje pri posredovanju paketa in ta skupina ni nujno najobetavnejša pri iskanju ciljnega vozlišča, lahko traja vzpostavljjanje poti do ciljnega vozlišča nekoliko dlje. Lahko se tudi pojavljajo *osamelci*, to je skupina, ki je ločena od ostalega dela omrežja in tako paket za vzpostavitev poti ne potuje nikamor. Pri tem Naumovem AODV se pojavlja tudi to, da lahko dve vozlišči znotraj skupine prejmeta in posredujeta paket naprej. To se da odpraviti s podatkom o naslovu predhodnega vozlišča.

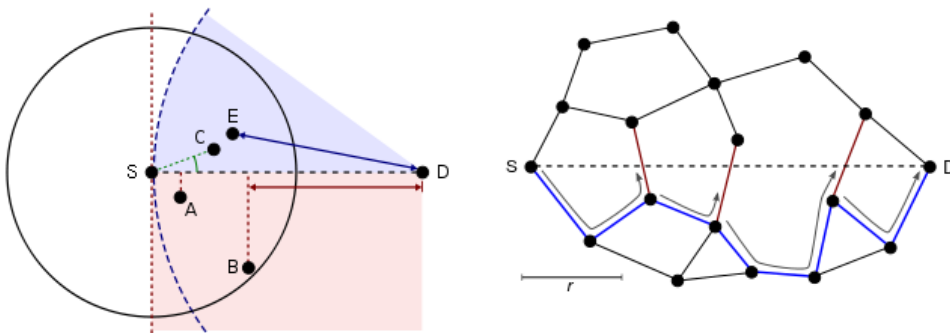
Razlika med DSR in AODV je v tem, da podatkovni paketi pri DSR vsebujejo informacijo o celotni poti do ciljnega vozlišča, medtem ko pri AODV paketi vsebujejo le naslov ciljnega vozlišča. Paketi DSR so tako večji in zahtevajo večjo pasovno širino omrežja. Tudi paketi *RouteReply* so različni pri obeh protokolih. Pri AODV vsebujejo informacijo o naslovu ciljnega vozlišča in podatek o tem kako nova je pot, pri DSR pa paketi vsebujejo naslove vseh vozlišč, ki so na poti do ciljnega vozlišča.

2.2 Geografsko usmerjanje

Position-based routing je usmerjanje na podlagi zavedanja avtomobila o lokaciji sosednjih vozil. Temelji na ideji da izvirno vozlišče pošlje podatkovne pakete na podlagi informacije o lokaciji ciljnega vozlišča¹ in ne na podlagi informacije o naslovu ciljnega vozlišča. Geografsko usmerjanje tako zahteva,

¹*beaconing* - vozlišče pošilja kratka pozdravna sporočila s katerimi izdaja svojo prisotnost in lokacijo

da ima vsako vozlišče, ki je del omrežja, podatek o svoji lokaciji in lokaciji ciljnega vozlišča. Podatkovni paket se pošlje brez vedenja o sestavi (topologiji) omrežja in brez predhodnega iskanja poti do ciljnega vozlišča. Obstaja več tehnik in pristopov pri implementaciji tega usmerjanja. Imamo pristop, kjer se uporablja ena sama pot do ciljnega vozlišča, pristop kjer se uporablja več poti do ciljnega vozlišča in tehniko poplavljanja omrežja.



Slika 2.2: Tehniki *Greedy forwarding* in *Face routing*

Pri strategiji uporabe ene poti do ciljnega vozlišča se uporablja tehniki *greedy forwarding* in *face routing*. Greedy forwarding poskuša paket poslati proti vozlišču D . Izvorno vozlišče S uporablja več možnih algoritmov za posredovanje paketov po omrežju. Ti so prikazani na sliki 2.2. Izvorno vozlišče posreduje pakete do sosednjega vozlišča. Najbolj ustrezno vozlišče je tisto, ki ima najkrajšo razdaljo E , imenovano *Greedy*, med vsemi vozlišči znotraj območja S , prikazanega na sliki 2.2. Lahko pa izbere med ostalimi vozlišči znotraj območja, vendar je vse odvisno od izbrane tehnike za izbiro sosednjega vozlišča. Pri uporabi nekaterih tehnik (NFP, Compass Routing) pa se izkaže, da se lahko pojavljajo tudi zanke.

Požrešno usmerjanje (angl. *Greedy routing*) ni povsem brez napak. Lahko se zgodi da vozlišče nima več sosedov, kateremu bi lahko poslal paket. Tu prav pride tehnika *face routing*, ki je prikazana na sliki 2.2. Ta tehnika pomaga, da paket najde pot do vozlišča, kjer se lahko naprej nadaljuje tehnika Greedy Forwarding.

Geografsko usmerjanje se deli na tri kategorije, *non-Delay*, *Delay* in *hibridno*. Non-Delay se uporablja v gosto poseljenem omrežju in za razliko od Delay (DTN - Delay Tolerant Network) ne predvideva delov omrežja, ki so nepovezani z ostalim delom omrežja. Hibridni način usmerjanja po omrežju je pa samo kombinacija non-Delay in Delay tehnike usmerjanja.

2.2.1 Non-DTN

Glavna stvar pri požrešnem pristopu (angl. *Greedy Forwarding*) je, da vozlišče posreduje paket tistemu sosednjemu vozlišču, ki je bližje ciljnemu vozlišču. Greedy Forwarding se lahko slabo izkaže, če nobeno sosednje vozlišče ni bližje kot trenutno vozlišče. V tem primeru je paket dosegel *lokalni maksimum* in prišel do *maksimuma* pri trenutnem vozlišču. Vsi usmerjevalni algoritmi v tej kategoriji imajo vsak svojo strategijo, da se izvlečejo iz take situacije.

2.2.2 GPSR

(Greedy Perimeter Stateless Routing) [11] posreduje paket sosednjemu vozlišču, ki je geografsko bližje ciljnemu vozlišču. Pri GPSR temu načinu pošiljanja rečemo *greedy mode*. Ko pri tem požrešnem načinu paket doseže lokalni maksimum, se GPSR izvleče iz tega položaja tako, da posreduje paket vozlišču, ki je bližje ciljnemu vozlišču. Temu načinu se reče *perimeter* način, ki uporablja desnoročno pravilo. Pošiljanje paketa nadaljuje v *greedy* načinu, dokler razdalja med vozliščem in ciljnim vozliščem ni manjša od razdalje med vozliščem, kjer je prišlo do lokalnega maksimuma, in ciljnim vozliščem.

Ko v desnoročnem pravilu vozlišče X doseže lokalni minimum, je njegovo naslednje vozlišče, kamor bo poslal paket, vozlišče Y. Vozlišče Y je izbrano zaradi desnoročnega pravila (navidezni trikotnik tvorijo vozlišče X, vozlišče Y in ciljno vozlišče D). Ta način posredovanja paketov se imenuje *face routing*. Tako ime pa zato, ker mora paket prepotovati t. i. *face* področja, ki ležijo na navidezni daljci med vozliščem X in ciljnim vozliščem D (prikazano na sliki

2.2). Pri *face routingu* pa se pojavljajo težave z zankami, zaradi katerih paket ne najde ciljnega vozlišča. To se odpravi z algoritmom RNG (angl. *Relative Neighbourhood Graph*) ali GG (angl. *Gabriel Graph*), ki navaden graf pretvorijo v ravninski ali planarni graf in take ne pride do zank znotraj grafa (omrežja).

Zaradi manjših pomanjkljivosti, kot so zastareli podatki v tabelah vozlišč o poziciji določenih vozlišč, je Naumov GPSR izboljšal z uvedbo AGF (angl. *Advanced Greedy Forwarding*). Tako sta smer in hitrost vsebovana v oddajnem signalu posameznega vozlišča, v podatkovnem paketu pa je poleg običajnega zapisano tudi celoten čas potovanja paketa, čas za obdelavo paketa in naslov vozlišča, ki trenutno posreduje paket. Vsako vozlišče lahko tako popravi svoje tabele o pozicijah sosednjih vozlišč in nima zastarelih podatkov, ki nastanejo zaradi premikajočih se vozlišč (vozil). Z informacijami v podatkovnem paketu pa lahko oceni dolžino do ciljnega vozlišča in njegovo trenutno pozicijo.

Izkaže se [12], da ima GPSR v primerjavi z DSR veliko boljši uspeh pri dostavljanju paketov do ciljnih vozlišč (Packet Delivery Ratio).

GPSR je protokol, ki izvira iz omrežij MANET in je velikokrat uporabljen kot izhodišče pri načrtovanju ostalih (in novih) usmerjevalnih protokolov za omrežja VANET.

2.2.3 PRB-DV

(Position-Based Routing with Distance Vector Recovery) vsebuje določene tehnike protokola AODV, vendar je vezan na pozicije vozlišč. V primeru da paket zaide do vozlišča, kjer ne more nikamor naprej (doseže lokalni maksimum), uporabi princip protokola AODV. Vozlišče pošlje zahtevek za pot do ciljnega vozlišča, v zahtevku sta pa vsebovana tudi trenutna lokacija vozlišča in lokacija ciljnega vozlišča. Ko vozlišče prejme paket z zahtevkom najprej preveri, če je bližje ciljnemu vozlišču kot vozlišče z lokalnim maksimumom. V primeru, da ni bližje, ponovi pošiljanje zahtevka in zapiše svojo lokacijo v paket, drugače pa pošlje odgovor nazaj proti vozlišču z lokalnim maksimumom.

mom, da naj proti njemu pošlje pakete. Ta del je podoben kot pri AODV. Ko potrditveni paket potuje nazaj do vozlišča z lokalnim maksimumom, se vmes vzdržuje pot (naslovi vozlišč), da lahko paketi kasneje pridejo do vozlišča, ki je bližje ciljnemu vozlišču.

Pri tem protokolu se izkaže [6] nepotrebna zasedenost omrežja v primeru, ko paketi dosežejo lokalni maksimum.

2.2.4 GRANT

(Greedy Routing with Abstract Neighbor Table) uporablja podobno kot GPSR *greedy routing*, vendar ga še nekoliko razširi. Vsakemu vozlišču doda še tabelo sosednjih vozlišč, ki so znotraj določenega območja. Območje je določeno s številom skokov od trenutnega vozlišča. Ker ima vpogled nekaj vozlišč naprej, paketi ne zaidejo do lokalnega maksimuma. Katero vozlišče bodo paketi izbrali kot naslednjega, pa je izračunano iz najkrajše razdalje med trenutnim vozliščem in potencialnim vozliščem ter število skokov, ki jih bodo morali opraviti paketi. Vozlišče, ki nudi najugodnejše izračune (najmanjši zmnožek prej naštetih vrednosti), je izbrano za naslednjo lokacijo (lokacijo kamor morajo potovati paketi). Pri GRANT pride znotraj izbranega območja preveč oddajnih signalov posameznega vozlišča in zato pride do previsokega prepletanja informacij o vozliščih. GRANT območje razdeli na polja in iz posameznega polja izbere samo enega predstavnika s katerim bo operiral. Polja pa si razdeli glede na prejeti oddajni signal vozlišča in oddajni signal njegovih sosedov, ter jih tako kategorizira.

Kljub izboljšanju ovire (lokalni maksimum), se izkaže [11] da je tradicionalni *greedy routing* boljši.

Naslednji protokoli so prekrivni protokoli (angl. *Overlay routing*), ki imajo nekaj drugačnih značilnosti od prej naštetih. Usmerjevalni protokoli tu operirajo prekrivnim grafom, ki je primerno organizirano (ima primerno topo-

logijo) za obdelavo. V realnem okolju se te operacije vidijo npr. v mestih, kjer potujejo paketi iz enega dela ceste na drug odsek ceste. Tako imajo ti protokoli opravka s stičišči vej v grafu, omrežju, kjer vozila spreminjajo smeri, torej križišči.

2.2.5 GPCR

(Greedy Perimeter Coordinator Routing) je Lochert opisal v [13] in pravi, da potreba po pretvorbi v ravninske grafe ni nujna. Ulice v mestih že same po sebi predstavljajo planarni graf. Vozila, ki predstavljajo vozlišča, v tem primeru posredujejo pakete po že znanem *greedy* in *perimeter* načinu vzdolž ceste, dokler ne pridejo na odsek ceste (navadno so to križišča), kjer ne vedo kam paket poslati naprej. Pri teh protokolih je torej pomembno, da se paketi pošiljajo tam, kjer vozila potujejo, torej po cestah. Pri GPCR paketi naredijo krajše skoke in odpravljene so napake, ki se pojavljajo pri pretvorbi v ravninske grafe. GPCR uporablja dve oceni za določitev naslednjega vozlišča. Prva ocena je sestavljena iz korelacijskega koeficienta, ki veže vozlišče na njegova sosednja vozlišča. Če je koeficient blizu 0, to pomeni, da ni linearne povezave med pozicijami sosednjih vozlišč, kar pa označuje da je vozlišče v križišču na zemljevidu. Torej je to vozlišče potencialni kandidat za posredovanje paketov do ciljnega vozlišča. Druga ocena določi ali je vozlišče X na križišču med dvema vozliščema Y in Z, če na podlagi informacije o lokaciji iz oddajnega signala ugotovi, da sta si vozlišča Y in Z v dosegu (preko vozlišča X), vendar nimata drug drugega v tabeli sosedov (angl. *List of Neighbors* je seznam vozlišč, ki so v dosegu enega skoka).

2.2.6 GPSR-J+

Protokol uporablja preskoke za pošiljanje paketov. Namesto da bi paket potoval od vozlišča do vozlišča po kakor pri GPCR, sedaj protokol uporabi predvidevanje za dva skoka naprej. Če trenutno vozlišče vidi, da bo paket spremenil smer potovanja, potem ga posreduje tistemu vozlišču, ki potuje v

tej drugi smeri, drugače paket posreduje naprej za dva skoka. Tako odpravi nepotrebne ustavitve na križiščih. Ko se znajde v *perimeter* načinu, uporabi desnoročno pravilo za določanje najboljše smeri. Če obstaja v tej smeri vozlišče z enako smerjo in je še vedno v dosegu, potem izbere njega, drugače posreduje paket vozlišču, ki ga je na podlagi dveh ocen iz GPCR izbral za vozlišče na križišču (vozlišče je povezano z dvema vozlišči). Od GPSR-J+ tako pričakujemo, da ima veliko boljši odziv v situacijah, ko paket doseže lokalni maksimum, kakor pa GPSR.

2.2.7 CAR

(Connectivity-Aware Routing) je Naumov predstavil v [14] in s pomočjo PGB (Preferred Group Broadcast) iz AODV za zmanjšanje paketov po omrežju za iskanje poti do ciljnega vozlišča in AGF (Advanced Greedy Forwarding) iz GPSR za upoštevanje hitrosti in smeri posameznega vozlišča. Novost pri tem protokolu so fiksne sidrane točke t. i. *anchor points*, ki se uporabijo pri beleženju poti do ciljnega vozlišča. Namesto uporabe vozlišča, ki se pojavljajo na poti potrditvenega paketa za pot, se sedaj uporabljajo samo vozlišča, ki so bližje križišču ali cestnemu zavoju. Protokol tako izkoristi osnovne topološke značilnosti tega naravnega omrežja. V paket za odkrivanje poti do ciljnega vozlišča je vključeno veliko manj vozlišč kakor običajno. Vozlišče se določi za fiksno točko, če smer vozlišča ni vzporedna s trenutno smerjo, ki je zapisana v paketu za določanje poti do ciljnega vozlišča. Ciljno vozlišče se na podlagi prejetih paketov odloči za najustreznejšo pot. Zanj najboljša je tista pot, po kateri podatkovni paketi hitreje pridejo in je bolj povezljiva. Nato s pomočjo AGF potrditveni paket potuje nazaj do izvirnega vozlišča. CAR se ne zanaša na zemljevid in uporablja proaktivno iskanje fiksnih točk.

Ko izvirno vozlišče prejme potrditveni paket, pošlje podatkovne pakete po fiksnih točkah do ciljnega vozlišča. Naslovi fiksnih točk so zapisani v potrditvenem paketu in lahko podatkovne pakete na način AGF posreduje preko teh vozlišč. CAR ima še nekaj varovalk za mobilna vozlišča. Da ne izgubi

ciljnega vozlišča uporabi *varuhe* (angl. *guards*). Ti varuhi pomagajo slediti informaciji o ciljnemu vozlišču in imajo še druge naloge, kot so posredovanje informacij o lokaciji ali pa celo preusmerjajo pakete.

2.2.8 GSR

(Geographic Source Routing) [15] uporablja algoritem Dijkstra za izračun najkrajše poti [16] do končnega vozlišča. Operira na prekrivnem grafu, kjer so vozlišča križišča na zemljevidu, povezave pa ceste, ki povezujejo ta križišča. Podatkovni paketi so tako poslani po poti, ki je sestavljena iz teh vozlišč, ki vodijo do ciljnega vozlišča. Dobra lastnost protokola GSR je ta, da se v situaciji, ko ni povezave med dvema križiščema, dobro odreže. Enostavno uporabi požrešno posredovanje paketov (angl. *greedy forwarding*).

2.2.9 A-STAR

(Anchor-Based Street and Traffic Aware Routing) [17] je že iz imena razvidno, da uporablja fiksne točke za povezavo do ciljnega vozlišča. Podobno kot GSR opravi pošiljanje paketov na prekrivnem grafu preko teh fiksnih točk. Ker pa se A-STAR zaveda prometa na cestah, se na podlagi gostote prometa odloči ali bo vključil fiksne točke v najkrajšo pot ali ne. Poti do končnih vozlišč so zasnovane na dveh vrstah prekrivnih grafov. Prvi graf je narejen s pomočjo realnega (trenutnega) stanja na cestah. To vzdržuje s pomočjo obcestnih enot (angl. *road-side units*), ki vsem vozilom sporočajo kakšno je stanje na cestah (gostota prometa). Ta graf se imenuje tudi dinamično ocenjen graf. Drugi statistično ocenjen graf je sestavljen iz prekrivnega zemljevida, ki predstavlja glavne prometne povezave, kjer imamo zagotovljen promet in s tem povezljivost med vozlišči. S pomočjo Dijkstre izračunamo najkrajšo pot na znanih podatkih iz tega statistično ocenjenega zemljevida.

V primeru, da se paketi tega protokola znajdejo v delu omrežja, ki ni povezan, to situacijo rešijo na poseben način. Na novo se izračuna fiksna točka, preko katere se posreduje pakete. Odsek poti, kjer so paketi zašli v

slepo ulico, pa se označi za nedelujočega z *out of service*. Vozlišča si tako popravijo naslove do ciljnega vozlišča, da ne pride do zastajanja paketov.

2.2.10 STBR

(Street Topology Based Routing) [18] izboljša samo povezljivost vozlišč med križišči. To naredi na sledeči način: na samem križišču se določi gospodarja (angl. *master*), ki skrbi za informacijo o tem ali obstaja povezava do naslednjega križišča na poti proti ciljnemu vozlišču ali ne. Gospodarji si tako izmenjuje informacije o povezavah med sosednjimi vozlišči. Gospodar ima tako dvostopenjsko tabelo o sosednih vozliščih: Prvi nivo predstavlja direktne povezave sosednih vozlišč z vozliščem, ki je na križišču in kateri se uporablja pri načrtovanju poti do ciljnega vozlišča. Drugi nivo so pa direktne povezave iz sosednjih vozlišč, ki so na križiščih, do vozlišč, ki so na križišču kjer so gospodarji.

Od A-STAR ali GSR se razlikuje v tem, da se paketi usmerjajo glede na geografsko razdaljo do ceste kjer je ciljno vozlišče, medtem ko se pri A-STAR ali GSR računa pot do ciljnega vozlišča s pomočjo algoritma Dijkstra za računanje najkrajše poti do ciljnega vozlišča.

2.2.11 GyTAR

(Greedy Traffic Aware Routing) [19] uporablja požrešen (angl. *greedy*) pristop za posredovanje paketov med križišči, kakor ostali zgoraj omenjeni protokoli v tej kategoriji. Ta protokol se od ostalih razlikuje v tem, da uporabi obcestne enote, s pomočjo katerih dobi število vozil na določenem odseku ceste in tako določi, kako povezljiv je ta del poti. Gostoto prometa in razdajo do ciljnega vozlišča posreduje vozliščem na križiščih, da se s pomočjo teh parametrov zavedajo povezljivosti med križišči. Odsekom poti se s pomočjo teh dveh parametrov določi utež, na podlagi katere se načrtuje pot do ciljnega vozlišča. GyTAR prav tako uporablja algoritem Dijkstra za izračun najkrajše poti do ciljnega vozlišča.

2.2.12 LOUVRE

(Landmark Overlays for Urban Vehicular Routing Environments) je združene zmožnosti protokolov za usmerjanje GSR in A-STAR. Lee je v [20] predstavil razdelitev prekrivnih protokolov v dve skupini. Prva skupina uporablja zmožnosti protokola STBR, kjer se vozlišča določajo na podlagi razdalje sosednjih vozlišč do ciljnih vozlišč in zmožnosti GyTAR, kjer se upošteva promet na določenih odsekih poti. Druga skupina pa uporablja proaktivni pristop kakor A-STAR ali GSR za določanje katera vozlišča bodo uporabljena v prekrivnem omrežju. Druga skupina tako vnaprej določa seznam vozlišč za pot do ciljnega vozlišča.

LOUVRE spada v drugo skupino in je zanimiv predvsem zaradi uspešnosti pri načrtovanju poti preko prekrivnih vozlišč. Protokol se zaveda dejstva, da ko promet doseže določeno gostoto, ostajajo povezave na prekrivnem omrežju, ki ne izgubijo povezljivosti. Mejo določi na podlagi oddajnih signalov na odsekih cest. Za izračun najkrajše poti do končnega vozlišča se uporablja algoritem Dijkstra. LOUVRE sedaj ne potrebuje obcestnih enot za ugotavljanje gostote prometa. Dobre lastnosti protokola so krajši čas za vzpostavitev poti do ciljnega vozlišča in redke pojave situacije lokalnega maksimuma. Slaba lastnost pa je uporaba tega protokola v prometnih scenarijih, kjer gostota prometa ni tako velika.

2.2.13 CBF

(Contention-Based Forwarding)[21] po razdelitvi na Sliki 2.1 spada v posebno kategorijo znotraj geografsko usmerjenih protokolov. Ta protokol ne uporablja pozdravnih sporočil sosednjih vozlišč ampak direktno pošlje podatkovne pakete. Katero pa bo izbrano naslednje vozlišče, pa je izbrano s pomočjo porazdelitvene funkcije (uporablja pristranski časovnik za izbor vozlišča), ki temelji na dejanskih pozicijah sosednjih vozlišč. Prvo izbrano vozlišče prepreči izbor ostalih sosednjih vozlišč. Tako ne pride do podvajanja paketov. Paketi napredujejo proti ciljnemu vozlišču in s tem se krajša tudi časovnik,

ki je uporabljen pri izbiri vozlišč.

2.2.14 TO-GO

(Topology-assist Geo-Opportunistic Routing)[22] spada v posebno podkategorijo geografskega usmerjanja. Protokol uporablja princip dveh skokov podobno kot GPSR-J+ za preskok določenih vozlišč. V *greedy* načinu skuša preskočiti vsako vozlišče znotraj križišča. TO-GO vpelje oportunistično funkcijo za posredovanje paketov. Vozlišča sedaj pakete posredujejo množici vozlišč. Množica vozlišč se poteguje za kandidata, ki bo naslednji posredoval pakete. V množici vozlišč je prvi kandidat tisti, ki je geografsko bližje ciljnemu vozlišču. Temu pravimo tudi prioriteto razvrščanje (angl. *Priority Scheduling*).

Od CBF se razlikuje po tem, da pri pošiljanju paketov po omrežju prioriteto izbira vozlišča, ki so pomembna za napredovanje paketov v *greedy* ali *perimeter* načinu. S tem TO-GO predvideva, da se večina scenarijev dogaja v mestih, kjer morajo paketi spremeniti smeri potovanja in ne na avtocestah. Za razliko od CRF uporablja ta protokol pozdravna sporočila znotraj množice vozlišč, da se med njimi lahko izbere primerne kandidata za oportunistično napredovanje paketa.

2.2.15 DTN kategorija

V kategorijo *Delay Tolerant Network* spadajo protokoli, ki uporabljajo strategijo nesi in posreduj (angl. *Carry-and-Forward*). To strategijo uporabljajo zaradi pogoste nepovezanosti med vozlišči. To se v mestih z nizko gostoto prometa pogosto dogaja. Omenjena strategija omogoča vozliščem, da shranijo pakete, ko nimajo povezave z drugimi vozlišči in jih po določenem času, pod določenimi pogoji posredujejo vozliščem, ki jih srečajo kasneje na poti.

2.2.16 VADD

(Vehicle-Assisted Data Delivery) [23] spada pod kategorijo DTN in uporablja nesi in posreduje strategijo s pomočjo značilnih lastnosti mobilnega omrežja, kot je na primer predvidljiva smer in hitrost vožnje. Pot za dostavo paketov je sestavljena iz križišč. Strategija odločanja na križišču o vozlišču za nadaljevanje poti je preprosta. Izbere vozlišče, ki ima najkrajši čas dostave paketa (angl. *Packet Delivery Delay*) do ciljnega vozlišča. Dolžina poti, gostota prometa na poti in povprečna hitrost vozila so parametri, ki jih protokol uporabi za določanje dostavnega časa paketa. Zhao je v [23] opisal veliko različic (L-VADD, D-VADD, MD-VADD, H-VADD) protokola VADD, ki pa se predvsem razlikujejo po strategiji izbire vozlišča na križiščih.

2.2.17 GeOpps

(Geographical Opportunistic Routing) [24] je prav tako *carry-and-forward* protokol in izbira med vozili, ki imajo večjo verjetnost za smer proti ciljnemu vozlišču.

Protokol izračuna razdaljo med ciljnim vozliščem in najbližjim odsekom poti. Izračuna tudi dostavni čas paketa do končnega vozlišča. Izbere med več možnimi potmi in se odloči za tisto, ki ga vodi bližje in hitreje k ciljnemu vozlišču. Med vožnjo vozila se lahko zgodi, da sreča vozilo, ki ima krajši izračunan dostavni čas. V tem primeru so paketi posredovani obetavnejšemu vozilu.

2.2.18 GeoDTN

(Geographical DTN + Nav) [25] je protokol, ki uporablja različne metode *non-DTN* in *DTN* pristopa (greedy način, primeter način in *carry-and-forward* način). Uporablja virtualni navigacijski vmesnik (angl. *Virtual Navigation Interface*) za pridobitev informacij, ki jih uporabi za izbiro načina in posredno za izbiro vozlišča, ki bo posredovalo paket proti ciljnemu vozlišču. Ker je pri GeOpps informacija o smeri vozil javna, je ta težava pri GeoDTN

rešena z uporabo navigacijskih vmesnikov.

2.2.19 RIVER

(Reliable Inter-VEhicular Routing) [26] je Bernsen razvil konec leta 2011. Protokol uporablja veliko prvin že obstoječih usmerjevalnih protokolov [11], [17]. Vozlišča (vozila) so opremljena z informacijami o trenutni lokaciji ter oddajnikom *beacon* sporočil za obveščanje sosednjih vozlišč o svoji prisotnosti (kot GPSR). Za pošiljanje sporočil do želenega cilja, uporablja požrešno posredovanje (Greedy forwarding - opisan v podrazdelku 2.2.2). Ima tudi komponento nadziranja prometa, za katero je potreben predhodno shranjen zemljevid cest in vozlišč (zemljevid vsebuje informacije o lokacijah križišč oziroma cestnih zavojev, ter podatke o tem kako so posamezna križišča med seboj povezana). Pakete poskuša zato poslati vzdolž poti, saj se zaveda, da le tam potujejo vozlišča (vozila) in se tako izogne nepotrebnemu zavračanju paketov (Paket je zavržen, ko ne najde vozlišča, kateremu bi posredoval paket).

Komponenta nadziranja prometa je realizirana s pošiljanjem sonar sporočil (angl. *Probe messages*) in s pasivnim nadziranjem ostalih sporočil, ki se pošiljajo po omrežju.

Poglavje 3

Omrežni Simulatorji

3.1 Splošno

Omrežni simulator je program, ki predvideva obnašanje omrežja. Simulatorje uporabljamo, ker bi bila simulacija na resničnih omrežjih časovno in materialno predraga. Uporabni so za protokole WLAN, UDP, TCP, v našem primeru VANET in podobno. Simulatorji nimajo možnosti v celoti posnemati delovanje realnega omrežja. Zato pa se s pravimi nastavitvami lahko dovolj dobro približamo delovanju pravega omrežja. Testiranja in analize simulacij nam pokažejo, kako bi v realnosti delovalo omrežje. Z vložki sprememb, ki se pojavljajo v realnem okolju, lahko s simulacijami spremljamo obnašanje omrežja. Simulatorji dajo uporabnikom na razpolago veliko tehnologij, s katerimi lahko zgradijo od preprostih omrežij, do zapletenih omrežij vozil, ki se pojavljajo v mestih. Obstaja veliko dostopnih omrežnih simulatorjev, ki so plačljivi ali pa prosto dostopni.

Spodaj bomo opisali nekaj predstavnikov omrežnih simulacij: NetSim, OPNET, OMNET++ in NS-2/NS-3 [27].

3.1.1 NetSim

Tetcos NetSim je izšel leta 2002 in ga lahko dobimo v standardni in akademski obliki. Namenjen je za operacijske sisteme Windows. NetSim je program

namenjen raziskavam računalniških omrežij. Napisan je v C in programskem jeziku Java. V knjižnicah ima vgrajene tehnologije WLAN, TCP, IP, UDP, Zigbee, MANET in podobno. V NetSim lahko spreminjamo že napisane knjižnice za naše potrebe. Razhroščevanje (angl. *debugging*) je deloma vključeno v simulator. Večinoma se uporablja v univerzitetnih inštitucijah (Anna University - Syria, Szczecin University - Poland, Barry University - USA, itd.).

3.1.2 OPNET

OPNET (OPNET Modeler) je plačljiv program, ki ponuja analizo računalniških omrežij in aplikacij. Zaradi komercialne narave daje uporabnikom bogat nabor orodij in široko grafično podobo. Napisan je v programskem jeziku C++. Omogoča modeliranje vseh možnih omrežij in ima vključene že nekatere tehnologije (MPLS, IPv6, in podobno).

3.1.3 OMNET++

Omrežni simulator OMNET++ je objektno usmerjen simulator in ima modularno razvojno okolje za izdelavo simulacij, ki bazirajo na računalniških in drugih omrežjih. Temelji na razvojnem okolju ECLIPSE (IDE - interaktivno razvojno okolje (angl. *Interactive Development Enviroment*)) Eclipse). Napisan je v programskem jeziku C++. Namenjen je za sisteme Windows, Linux, Unix in OS X. Za izdelavo simulacij uporablja visoko-nivojski programski jezik Ned. Pred začetkom simulacije je potrebno nastaviti ustrezne parametre v nastavitveni datoteki. Simulacije dajo izhodne datoteke, ki se lahko uporabijo v matematičnih programih Matlab ali R. S pomočjo teh programov lahko prikažemo, obdelamo in analiziramo dobljene rezultate.

3.1.4 NS-2

NS-2 diskretni simulator za izvajanje omrežnih simulacij. Odprto-kodni simulator lahko simulira vse omrežne funkcije in protokole s pomočjo različnih

modulov. Nastal je leta 1989. NS-3 je naslednik NS-2 in je narejen kot samostojni projekt. Ker ni nadgradnja NS-2, ne podpira simulacij, ki jih lahko izvajamo v NS-2. Uporabniki lahko implementirajo objekte v programskem jeziku C++ ali Python.

Omrežni simulator NS-2 je najbolj razširjen odprto-kodni omrežni simulator. Razvit je bil leta 1989 kot del projekta VINET [28]. NS-2 je objektno orientiran simulator časovno diskretnih dogodkov, ki jih ureja razvrščevalnik (angl. *scheduler*) dogodkov. Program je zelo uporaben za izvajanje manjših in večjih simulacij računalniškega omrežja. Namenjen je simuliranju žičnega, brezžičnega in tudi kombiniranega omrežja (angl. *wired-wireless network*). Zaradi svoje modularnosti in preprostosti je pridobil veliko število uporabnikov, ki lahko poljubno spreminjajo že obstoječo izvorno kodo simulatorja in ji dodajajo novo.

NS-2 simulator omogoča implementacijo simulacijskih scenarijev imenovanih tudi simulacijske skripte (angl. *simulation scripts*). Skripte so napisane v programskem jeziku OTcl (*Object-oriented Tool Command Language*) in programskem jeziku C++. Programski jezik C++ skrbi za zahtevno funkcionalnost pri izvajanju. Veliko uporabnih funkcij (mobilna vozlišča, agenti posamezne plasti, povezave med vozlišči, vrste za pošiljanje paketov, aplikacijska omrežna plast) je že vgrajenih v omrežni simulator. Ostale manjkajoče funkcionalnosti, kot so omrežni protokoli, pa lahko uporabnik poljubno implementira in vgradi v obstoječe okolje simulatorja NS-2. V sam simulator se lahko vključi različne vzorce in promet omrežja do potankosti. Vključi se lahko tudi napake, ki se pojavljajo v računalniških omrežjih, tako da se simulacije čimbolj približajo realnemu stanju.

Simulacije se zapišejo v *trace* datoteke, katere lahko obdelamo z različnimi orodji za analizo. Sledilne datoteke dajejo informacije o paketih in celotnem dogajanju v računalniškem omrežju, ki ga simuliramo. Poleg sledilnih datotek pa lahko ustvarimo *nam* datoteke, ki jih uporablja *Network Animator*. NAM omogoča vizualno predstavitev omrežne simulacije in je del NS-2 simulatorja. Za boljši prikaz rezultatov simulacije lahko uporabimo tudi GNUPlot

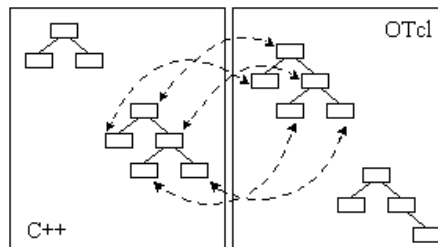
za risanje grafov.

Poglavje 4

Arhitektura NS-2

4.1 Dualnost OTcl in C++ okolja

Za izboljšanje kakovosti, fleksibilnosti in učinkovitosti se v omrežnem simulatorju NS-2 uporabljata dva programska jezika, OTcl in C++. OTcl je interpreter, zato ne potrebujemo prevajanja ob vsaki spremembi v izvorni kodi programa.



Slika 4.1: Povezljivost programskih jezikov OTcl in C++

Programski jezik OTcl uporablja sintakso, ki omogoča uporabo simulacijskih modelov. OTcl se uporablja za *raziskovanje zmogljivosti omrežij*, za *načrtovanje topologije omrežja*, za *omrežne scenarije*, za *manjše omrežne protokole* in *implementacijo glavne kode NS-2*. Za določena opravila je OTcl prepočasen, za kar se uporabi programski jezik C++. Ta se večinoma upo-

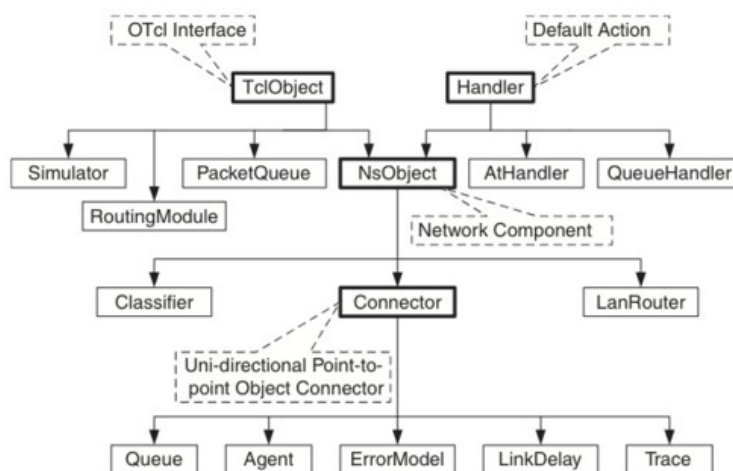
rablja za procesiranje paketov in upravljanje z dogodki.

Dualnost teh dveh programskih jezikov omogoča, da so lahko C++ objekti dosegljivi iz OTcl okolja (in obratno). Povezava ustvari OTcl objekte dostopne za C++ objekte in omogoči dostop do spremenljivk C++ objektov. Ker je omogočen dostop iz C++ kode do OTcl interpreterja, se na ta način lahko implementirajo komponente v C++ in OTcl.

Povezljivost med OTcl in C++ uporabniku daje transparentnost pri pisanju simulacijskih scenarijev in izbiri programskega jezika za implementacijo.

4.2 Razredna hierarhija v NS-2

Hierarhija razredov je prikazana na Sliki 4.2. Za boljše razumevanje značilnosti komponent omrežja, bomo v nadaljevanju na kratko opisali to strukturo.



Slika 4.2: 6 kategorij razredne hierarhije v NS-2

Koren ureditve je razred *TclObject*, ki je nadrazred vsem OTcl knjižnicam kot so omrežne komponente, razvrščevalniki in drugi objekti. *NsObject* je nadrazred vsem objektom, ki imajo opravka s paketi po omrežju (vozlišča, povezave in podobni objekti). Glavne komponente omrežja so razdeljene na dva podrazreda, *Connector* in *Classifier*. Razred *Classifier* vsebuje objekte,

ki imajo možnih več izhodnih podatkovnih poti. Razred *Connector* pa vsebuje glavne objekte omrežnih komponent, ki imajo samo eno izhodno podatkovno pot.

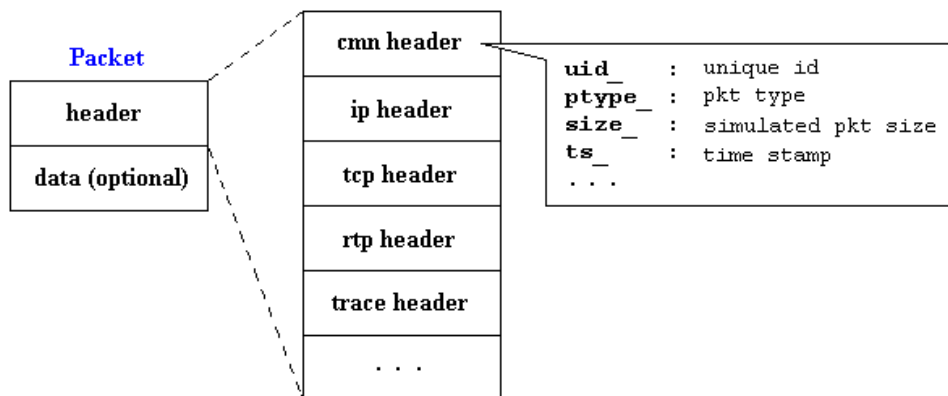
4.3 Dogodki v NS-2

Diskretni časovni razvrščevalnik (angl. *Discrete Event Scheduler*) skrbi za razvrščanje dogodkov v NS-2. Dogodke ustvarimo v napisanih simulacijskih skriptah ali pa se ustvarijo v omrežnih komponentah, ki skrbijo za zamik obdelave paketov ali pa nastanejo pri uporabi t. i. časovnikov (angl. *timers*).

NS-2 pozna dva razvrščevalnika dogodkov. Prvi skrbi za interakcijo med simulatorjem in pravim omrežjem v realnem času - *real-time scheduler*. Drugi *non-real-time scheduler* pozna tri tipe razvrščevalnikov, privzeti koledar (angl. *Calendar*), seznam (angl. *List*) in kopico (angl. *Heap*).

Razvrščevalniki skrbijo za zapis časa posameznega dogodka, njegovo identiteto (angl. ID) in rokovalnik (angl. Handler). Dogodkovni razvrščevalnik je v NS-2 posredno dosegljiv pri uporabi časovnikov, ki jih bomo opisali kasneje, ali pa preko instance OTcl interpreterja. V simulacijskem scenariju je to ukaz "at <čas ><ukaz >".

4.4 Paketi



Slika 4.3: Paketni format v NS-2

Paket v NS-2 je sestavljen iz *glave* (sklad) in opcijsko tudi prostora za podatke (prikazano na Slika 4.3). Paket oziroma format (angl. *Packet Header Format*) se inicializira, ko ustvarimo objekt *Simulator*. Ob tem se definirajo vsa definirana zaglavja (angl. *Header*), največkrat t. i. *common header*. Definirajo se lahko tudi *IP Header*, *trace header* in podobni. Ker pa je format paketa predstavljen kot sklad, se mora zapisati tudi odmik, da se pri dostopu ve, kje se posamezna glava nahaja. Ko *Agent* alocira paket, je uporabljena izbrana glava in sestavi se sklad iz registriranih zaglavij. Tako lahko posamezen omrežni objekt lahko dostopa do izbrane glave z uporabo ustreznega odmika.

4.4.1 Tipi paketov in glave

V NS-2 so posamezni tipi paketov definirani v datoteki *packet.h* (nahaja se v mapi *common*¹). Ko začnemo simulacijo razred *PacketHeaderManaged*

¹Dejanski naslov je `~ns/common/`, kjer `~` naslov do mesta, kjer smo namestili program NS-2

pregleda vse razpoložljive tipe paketov in vsakemu paketu dodeli odmike za posamezen tip paketa. Do omika dostopamo z metodo `access()`.

Koda 1 Paketni tipi definirani v `packet.h`.

```
enum packet_t {
    PT_TCP,
    ...
    PT_NTTYPE // This MUST be the LAST one
};

class p_info {
public:
    p_info() {
        name_[PT_TCP]= "tcp";
        ...
    }
}
```

Pri velikih simulacijah nam definiranje tipov za vse pakete upočasni izvajanje. Da preprečimo nepotrebno porabo virov (največkrat pomnilnik) v simulacijski skripti uporabimo ukaz `remove-packet-header`.

Najpomembnejša glava v NS-2 simulatorju je *common header*. Uporablja se za sledenje meritvam in dostopanje do podatkov pomembnih pri izvajanju simulacije.

V `hdr_cmn` - *common header* `p_type_` označuje tip paketa in služi za lažje sledenje med simulacijo. Postavka `uid_` je unikatna številka ID in jo uporablja razvrščevalnik, da si pomaga pri razvrščanju paketov. Polje `size_` pove velikost paketa v bajtih in je skupaj s podatki in velikostjo glav ostalih nivojev uporabljeno pri izračunu časa, ki ga bo paket potreboval da pride do določenega vozlišča. Polje `ts_` se uporablja za izračuna zamika, ko je posamezen paket v vrsti. Polje `error_` in polje `errbitcnt_` sta namenjena zaznavi ali je paket poškodovan in koliko bitov je poškodovanih. Polja `direction_`, `prev_hop_`,

`next_hop_` in `num_forwards_` so pomembna pri brezžičnih omrežjih. Polje `iface_` pa se uporablja pri t. i. *multicast* distribuciji po omrežju.

Klic ukaza `alloc()` rezervira pomnilnik za nov paket in vrne kazalec na to rezervirano lokacijo. S klicem ukaza `free()` se kazalec na lokacijo zbriše, podatki o paketu pa še vedno ostanejo in so del seznama prostih paketov. Obe metodi sta del razreda `Packet`.

4.4.2 Metodi *send* in *recv*

V NS-2 pošiljanje in sprejemanje paketov poteka na način, da se omrežne komponente povezujejo med seboj. Dostavljanje paketov je omogočeno preko povezav med vozlišči. Agenti generirajo pakete in jih pošljejo preko povezav do drugih vozlišč. Vozlišča, ki jih bomo uporabili za pošiljanje paketov, so navedena v simulacijski skripti.

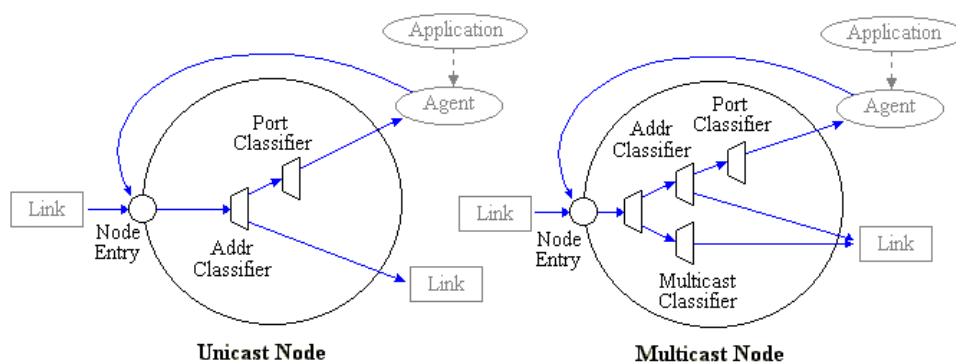
Koda 2 Omrežna komponenta (objekt) pošlje paket s `send` metodo, pri klicu pa vzbudi `recv` metodo kličočega objekta.

```
send(Packet *p, Handler *h) {  
    target_ = recv(p, h);  
}
```

4.5 Vozlišča

Vozlišča so glavne entitete izmed vseh omrežnih komponent v NS-2. Izvajajo procesiranje in posredovanje paketov, kar je glavnega pomena pri implementaciji usmerjevalnega protokola. NS-2 pozna ožičena vozlišča, brezžična vozlišča in kombinacijo le teh. Več podrobnosti se lahko najde v dokumentaciji omrežnega simulatorja NS-2 [29]. V nadaljevanju se bomo po večini posvetili vozliščem, ki se uporabljajo v brezžičnih mobilnih omrežjih.

4.5.1 Ožičeno vozlišče



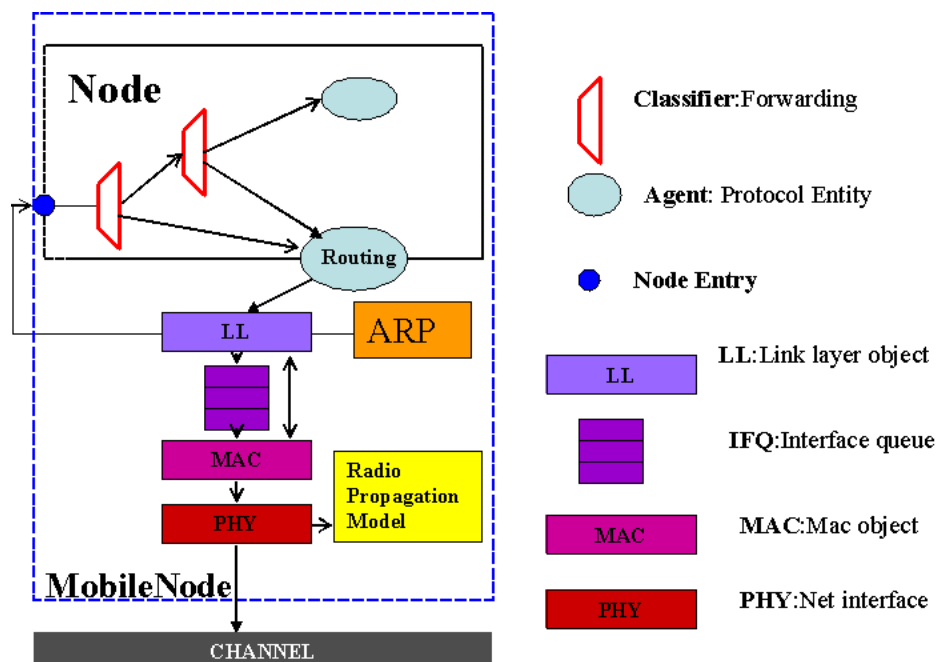
Slika 4.4: Ožičeno vozlišče (*unicast* in *multicast*)

Za boljšo predstavo bomo omenili tudi ožičeno vozlišče, ki je prikazano na Sliki 4.4. Vozlišče je sestavljeno iz vstopnega dela vozlišča in dveh razvrščevalnikov. Paketi najprej pridejo na vhodni del vozlišča. Naslovni razvrščevalnik (angl. *Address Classifier*) pregleda naslovni del paketa in ugotovi ali je ta paket namenjen trenutnemu vozlišču. Portni razvrščevalnik (angl. *Port Classifier*) odloči kateremu protokolu (agentu) znotraj vozlišča naj posreduje paket.

V primeru, da paketi niso namenjeni trenutnemu vozlišču, naslovni razvrščevalnik pakete posreduje pravemu vozlišču preko povezave (angl. *Link*).

NS-2 pozna dva tipa vozlišč glede na vrsto razpošiljanja paketov. Vozlišče *unicast* ima naslovni razvrščevalnik, ki opravi *unicast* razvrščanje, in portni razvrščevalnik. *Multicast* vozlišče ima poleg gradnikov *unicast* vozlišča, tudi razvrščevalnik, ki loči *multicast* pakete in *unicast* pakete. Ima tudi *multicast* razvrščevalnik (angl. *Multicast Classifier*), ki opravi nalogo *multicast* razvrščanja.

4.5.2 Mobilno vozlišče



Slika 4.5: Shema mobilnega vozlišča v NS-2

Mobilno vozlišče se od ožičenega vozlišča opisanega v podrazdelku 4.5.1 razlikuje v dodatnih funkcionalnostih, ki omogočajo prilagajanje mobilnim omrežjem. Razlika je tudi v povezavi mobilnega vozlišča z brezžičnim kanalom. Ožičeno vozlišče pri komunikaciji uporablja povezave. Mobilno vozlišče se lahko tudi premika znotraj topologije, medtem ko je ožičeno vozlišče stacionirano na določeno mesto.

Na Sliki 4.5 je mobilno vozlišče sestavljeno iz naslednjih delov:

- *Classifier*: Naslovni in portni razvrščevalnik skrbita za posredovanje paketov agentom. Privzeto je naslovljen usmerjevalni agent preko porta 255.
- *Routing*: Usmerjevalni agent ima nalogo vzdrževanja usmerjevalne tabele in posredovanja paketov. Paketu nastavi polje `next_hop`.

- *Link Layer*: Namenjen je pretvorbi omrežnih naslovov v strojne naslove (angl. *Hardware Addresses*) in pakete pripravi na to da bodo poslani na brezžični kanal. Pretvorbo naredi skupaj z ARP modelom.
- *ARP model*: Pomaga pri pretvorbi v MAC naslove.
- *IFQ*: Shrani pakete, ki bodo bili poslani na brezžični kanal, v vrsto.
- *MAC*: MAC nivo (angl. *MAC Layer*) upravlja z dostopom do brezžičnega kanala.
- *PHY*: Omrežni vmesnik pošilja in prejema pakete preko brezžičnega kanala.
- *Radio Propagation Model*: Oceni moč signala prejetih paketov in se odloči, ali jih omrežni vmesnik lahko prejme ali ne.
- *Channel*: Brezžični kanal preko katerega poteka pošiljanje/prejemanje paketov.

4.5.3 Konfiguracija in ustvarjanje vozlišč

Konfiguracija vozlišča je pomembna in se jo naredi pred ustvarjanjem vozlišča. To se naredi z ukazom `node-config`, ki ga izvede instanca simulatorja in posreduje nastavitve preko argumentov ukaza. Konfiguracija mobilnega vozlišča, ki se uporabi v simulacijski skripti je prikazana v Kodi 3, ostale možnosti nastavitvev pa so prikazane v Tabeli 4.1.

V primeru, da v simulacijskem scenariju ne opredelimo konfiguracije vozlišča, se uporabijo privzete nastavitve. Torej se uporabi ožičeno vozlišče z preprostim naslavljanjem od 0 naprej.

Vozlišča ustvarimo z ukazom `node`. Ustvari se vozlišče z lastnostmi, ki jih prej določimo z ukazom `node-config`. Ukaze lahko večkrat ponovimo z različnimi konfiguracijami.

Koda 3 Primer konfiguracije brezžičnega *ad-hoc* vozlišča.

```
$ns_ node-config -adhocRouting CAR \  
    -llType LL \  
    -macType Mac/802_11 \  
    -ifqType Queue/DropTail/PriQueue \  
    -ifqLen 50 \  
    -antType Antenna/OmniAntenna \  
    -propType Propagation/TwoRayGround \  
    -phyType Phy/WirelessPhy \  
    -channelType Channel/WirelessChannel \  
    -agentTrace ON \  
    -routerTrace ON \  
    -macTrace OFF \  
    -movementTrace ON
```

Tabela 4.1: Vse možnosti pri nastavitvah za vozlišče (`~ns/tcl/lib/ns-lib.tcl`).

Možnosti	Razpoložljive vrednosti	Privzeta vrednost
addressType	flat, hierarchical	flat
MPLS	ON, OFF	OFF
wiredRouting	ON, OFF	OFF
llType	LL, LL/Sat	-
macType	Mac/802_11, Mac/Csma/Ca, Mac/Sat, Mac/Sat/UnslottedAloha, Mac/Tdma	-
ifqType	Queue/DropTail, Queue/DropTail/PriQueue	-
phyType	Phy/WirelessPhy, Phy/Sat	-
adhocRouting	DIFFUSION/RATE, DIFFU- SION/PROB, DSDV, DSR, FLOODING, OMNIMCAST, AODV, TORA, M-DART, PUMA, <custom ad-hoc routing protocol >	-
propType	Propagation/TwoRayGround, Propagation/Shadowing	-
propInstance	Propagation/TwoRayGround, Propagation/Shadowing	-
antType	Antenna/OmniAntenna	-
channel	Channel/WirelessChannel, Channel/Sat	-
topoInstance	<datoteka s topologijo omrežja >	-
mobileIP	ON/OFF	OFF
energyModel	EnergyModel	-
initialEnergy	<vrednost v Joulih >	-
rxPower	<vrednost v W >	-
txPower	<vrednost v W >	-
idlePower	<vrednost v W >	-

Možnosti	Razpoložljive vrednosti	Privzeta vrednost
agentTrace	ON, OFF	OFF
routerTrace	ON, OFF	OFF
macTrace	ON, OFF	OFF
movementTrace	ON, OFF	OFF
errProc	UniformErrorProc	OFF
FECProc	?	?
toraDebug	ON, OFF	OFF
satNodeType	polar, geo, terminal, geo-repeater	-
downlinkBW	<velikost pasovne širine, Primer:"10Mb">	-

4.5.4 Pošiljanje in sprejemanje paketov

Naslavljanje vozlišč je sestavljeno iz 32-bitnega polja *node ID* in *port ID*. Skupaj tvorita naslov za dostavo paketov. V NS-2 obstajata tudi dva načina naslavljanja, ki jih nastavimo z ukazom `node-config`:

- *Ravninsko naslavljanje* (angl. *Flat Addressing*) - naslove se vozliščem dodeljuje ob njihovem nastanku. Podeljuje se jih po vrsti od 0 naprej. Naslovi lahko tudi služijo za štetje nastalih vozlišč.
- *Hierarhično naslavljanje* (angl. *Hierarchical Addressing*) - naslovi vozlišč se delijo po nivojih, kjer so nivoji določeni z določenim številom bitov.

Pošiljanje paketov poteka glede na Sliko 4.5 in Sliko 4.6 na sledeč način. Generator prometa (CBR, FTP), ki je pripet na ustreznega *Agent*a (UDP, TCP), generira pakete. Vsak generiran paket se preko omrežnega protokola dostavi na vhod mobilnega vozlišča. Nato naslovni razvrščevalnik preveri, komu je namenjen trenutni paket. Paket je lahko namenjen trenutnemu vozlišču, ali pa se posreduje naprej.

Posredovane pakete se dostavi usmerjevalnemu agentu in ta ustrezno obdela paket. Doda mu polje `next_hop` in paket posreduje *povezovalni plasti*. Povezovalna plast ima nalogo pretvoriti destinacijski naslov v MAC naslov. Pri pretvorbi pomaga ARP modul, ki generira ARP zahteve, procesira ARP odgovore in doda MAC glavo v paket. Paket se nato posreduje IFQ.

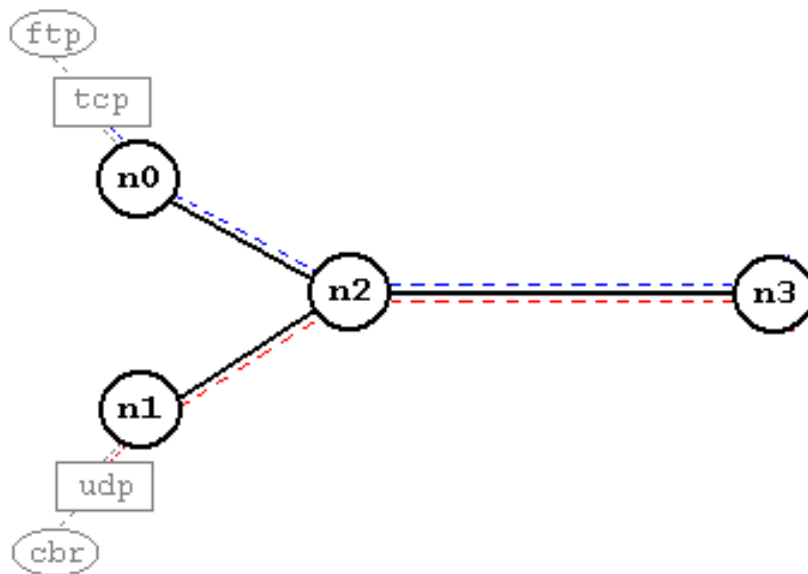
V *Interface Queue* čakajo paketi, ki bodo bili poslani na brezžični kanal. Velikost vrste je lahko omejena in paketi v njej so lahko razporejeni glede na določeno prioriteto.

Ko se sprostí brezžični kanal, MAC plast vzame iz vrste paket, in ga posreduje *fizični plasti*. Fizična plast postavi paket na brezžični kanal. Tako so paketi preko brezžičnega kanala dostavljeni fizičnim plastem ostalim mobilnim vozliščem.

Pri **prejemanju paketov** igra pomembno vlogo *radijski model razširjanja* (angl. *Radio Propagation Model* - RPM), ki določi ali je paket lahko pravilno prejet (brez napak) ali ne (z napakami). Na RPM vplivajo nastavitve, ki jih bomo omenili kasneje v razdelku.

Če je paket pravilno prejet, ga fizična plast posreduje MAC plasti. MAC plast ga potem posreduje povezovalni plasti, katera ga kasneje dostavi na vhod mobilnega vozlišča. Tam je spet posredovan naslovnemu razvrščevalniku, ki preveri če se naslov ujema z naslovom trenutnega vozlišča. Če se ujema je posredovan drugemu razvrščevalniku, ki preverja vrata (port classifier).

V primeru, da se naslov ne ujema s trenutnim vozliščem, je paket posredovan privzetemu agentu, ki je največkrat usmerjevalni agent. Usmerjevalni agent ima nato nalogo procesiranja paketa ali pa ga posredovanje do drugega mobilnega vozlišča. Razvrščevalec za vrata posreduje paket ustreznemu agentu mobilnega vozlišča (glede na podana vrata v paketu).



Slika 4.6: Primer preproste topologije v NS-2

4.5.5 Nastavitve v simulacijski skripti

Osnovne nastavitve so pomembne, da sploh lahko dobimo ustrezne rezultate in jih kasneje analiziramo. Omogočiti moramo sledenje za NAM (angl. *Network Animator*), ki beleži premike vozlišč in pošiljanje paketov. Definirati moramo topografijo mobilnega omrežja. To storimo na sledeč način:

Koda 4 S tem ustvarimo mrežo velikosti X x Y.

```
set topo [new Topography]
```

```
$topo load_flatgrid $x $y
```

Skupaj z opcijo `-topoInstance` v nastavitvah za vozlišče (`node-config`) je instanca topografije posredovana vozlišču.

Pomemben objekt, ki ga moramo definirati v simulacijski skripti, je GOD (angl. *General Operations Director*). Zgradimo ga po sledečem postopku:

Koda 5 Kreiranje objekta GOD.

```
set god_ [create-god $nn] %nn is the total number of nodes
```

Objekt GOD zbira informacije o optimalnem številu skokov od izvirnega vozlišča do ciljnega vozlišča. Ti podatki se uporabijo pri vrednotenju usmerjevalnih protokolov in statističnih podatkih.

Če želimo, da so kontrolni paketi našega usmerjevalnega protokola prej obdelani, lahko vrsti vmesnika nastavimo prioriteto:

Koda 6 Prioritiziranje kontrolnih paketov lastnega usmerjevalnega protokola.

```
Queue/DropTail/PriQueue set My_Own_Routing_Protocol 1
```

Nastavimo fizične lastnosti omrežnega vmesnika ter nastavimo plast MAC, da ustreza simulaciji.

Koda 7 Preprečimo naključno premikanje vozlišč.

```
set mobilenode_ [$ns_ node]
$mobilenode_ random-motion 0
```

Konfiguracija in kreiranje mobilnih vozlišč Mobilno vozlišče ustvarimo z ukazom `node`, ter nastavimo njegove lastnosti z že omenjenim ukazom `node-config`.

Nastavitve za premikanje vozlišč uporabimo, če želimo posnemati realno stanje premikanja vozlišč (Primer: premikanje vozil). Mobilna vozlišča beležijo svoje trenutne pozicije in se gibljejo znotraj topografije, ki jo nastavimo za simulacijo. Vozlišča se lahko gibljejo v vseh koordinatnih smereh, vendar je koordinata Z ponavadi ignorirana. S Kodo 8 nastavimo začetno pozicijo vozlišča.

Koda 8 Inicializacija pozicije mobilnega vozlišča.

```
$mobilenode_ set X_ 5.0
$mobilenode_ set Y_ 10.0
$mobilenode_ set Z_ 0.0
```

Če želimo simulacijo prikazati v NAM (angl. *Network Animator*), ne smemo pozabiti na nastavitve:

Koda 9 Nastavitve za NAM - pozicija in velikost prikazanega vozlišča.

```
$ns_ initial_node_pos $mobilenode_ 10
```

Koda 9 omogoči NAM, da pravilno pozicionira in izriše mobilno vozlišče. To naredi z ukazom `initial_node_pos <mobilno vozlišče> <velikost>`.

Koda 10 Definiranje premika mobilnega vozlišča.

```
$mobilenode_ setdest 10.0 20.0 5.0 %<x> <y> <velocity>
```

Mobilno vozlišče premikamo z ukazom `setdest`, kjer mu določimo destinacijo z koordinatami x in y , ter hitrostjo v m/s.

Koda 11 Premik v določenem času.

```
$ns_ at 1.0 $mobilenode_ setdest 10.0 20.0 5.0
%<x> <y> <velocity>
```

Če želimo mobilno vozlišče premakniti na določene koordinate v določenem trenutku, lahko uporabimo Kodo 11.

4.5.6 Usmerjanje po mobilnem omrežju

Usmerjevalni agent mobilnega vozlišča prevzame vso vlogo upravljanja s paketi (usmerjanje in posredovanje) in računanja poti do ciljnega vozlišča.

Samo mobilno ne vsebuje usmerjevalne tabele, ampak jo vzdržuje sam usmerjevalni agent interno. Trenutno ima NS-2 implementacije usmerjevalnih agentov najbolj značilnih usmerjevalnih protokolov za brezžična omrežja; DSDV, DSR, TORA, AODV in PUMA.

4.6 Agenti

Agenti so končne entitete vsakega omrežnega nivoja, kjer se paketi generirajo ali porabijo. Agenti služijo za implementacijo protokolov na različnih omrežnih plasteh. Primeri agentov, ki so pogosto uporabljeni pri simulacijah v NS-2:

- Usmerjevalni agenti (angl. *Routing Agents*)
- Ponori prometa (angl. *Traffic Sinks*)

Implementacijo agentov omogočata OTcl razred *Agent* in C++ razred *Agent*. Izvorno kodo razredov si lahko ogledamo v `agent.{cc, h}` in `ns-agent.tcl` (v mapi `common/` oziroma `tcl/lib/`).

4.6.1 Informacija o stanju agenta

Vsak agent vsebuje informacije, ki se jih podeli paketom še preden so poslani. Vrednosti, ki se dodelijo paketom, so prikazane v Tabeli 4.2.

Tabela 4.2: Informacije o stanju agenta (`~ns/agent.h`).

Spremenljivka	Opis
<code>addr</code>	Izvorni naslov trenutnega agenta
<code>dst</code>	Naslov kamor še pošilja paket
<code>size</code>	Velikost paketa v bajtih
<code>type</code>	Tip paketa
<code>fid</code>	ID IP toka
<code>prio</code>	Polje IP prioritete
<code>flags</code>	Zastavice paketa
<code>defttl</code>	Privzeta vrednost IP TTL

4.6.2 Metode in ukazi agentov

- `[]Packet* allocpkt` - določi prostor privzete velikosti za nov paket (shrani informacije o stanju agenta)
- `[int]Packet* allocpkt` - rezervira prostor velikosti n bajtov za nov paket in shrani informacije o stanju agenta
- `[Packet*, Handler*]void recv` - metoda za prejemanje paketov, ki so namenjeni agentu
- `[Packet*, Handler*]void send` - metoda s katero so paketi poslani

Agenti se dodelijo vozlišču z ukazom `attach-agent <vozlišče><agent>`. Ta ukaz nastavi agenta na portni razvrščevalnik s pomočjo polja `agent.port...`. Po dodelitvi lahko agent prejema vse pakete, ki mu jih posreduje portni razvrščevalnik na podlagi številke vrat agenta (angl. *Agent Port*).

4.6.3 Kreiranje novega agenta

Pri ustvarjanju novega usmerjevalnega agenta najprej podedujemo razred *Agent*. Za prejemanje paketov nato definiramo metodo `void recv(Packet`

*p, Handler *h). Za periodično proženje programske kode definiramo tudi časovnike. Koda 12 prikazuje povezavo agenta z OTcl okoljem. Mapiramo imenski prostor OTcl in vzpostavimo ime razreda za našega agenta.

Koda 12 Povezava z OTcl okoljem.

```
static class MyAgentClass : public TclClass {
public:
    MyAgentClass() : TclSclass("Agent/MyAgent") {}
    TclObject *create(int argc, const char*cosnt* argv) {
        return (new MyAgent());
    }
} class_myAgent;
```

Dostop do metod iz OTcl okolja omogočimo na naslednji način:

Koda 13 Metoda *bind* za povezavo metod iz OTcl okolja.

```
MyAgent::MyAgent() : Agent(PT_MYAGENT) {
    bind("packetSize_", &size_);  %accessing size_ from OTcl
}
```

Za prejemanje ukazov iz OTcl okolja ne smemo pozabiti povoziti metode `Agent::command()`:

Koda 14 Dodajanje ukaza zagon agenta.

```
int MyAgent::command(int argc, const char*const* argv)
{
    if (argc == 2) {
        if (strcmp(argv[1], "start") == 0) {
            startAgent();
            return TCL_OK;
        }
    }
    return (Agent::command(argc, argv));
}
```

4.7 CMU Monarch

Projekt Monarch[30] se je začel leta 1992 na Rice University - Department of Computer Science. S tem projektom so NS-2 omrežnem simulatorju dodali ogromno dodatkov, ki omogočajo zelo dobro simuliranje mobilnih vozlišč v brezžičnem okolju:

- Implementacija MAC protokola IEEE 802.11 DCF
- Implementacija ARP (Address Resolution Protocol)
- Implementacija *multi-hop* ad hoc omrežnih usmerjevalnih protokolov (AODV, DSR, DSDV, TORA)
- Podpora za nove plasti MAC
- Analiza sledilnih datotek
- Modeliranje radijskega signala (*Two Ray Ground Reflection radio propagation model*)
- in še veliko drugih uporabnih dodatkov za NS-2.

4.8 Sledilne datoteke - *Trace Files*

Obstaja veliko načinov za zbiranje izhodnih podatkov, ki jih kasneje lahko analiziramo. Eden od načinov je zbiranje sledi (angl. *Traces*), ki beleži vsak korak paketa. Beleži kdaj prispe paket, kdaj zapusti določeno mobilno vozlišče in kdaj je paket zavržen. Sledilni objekti so umeščeni v topografijo omrežja kot vozlišča. Ponavadi so pripeti ostalim objektom kot ciljni naslov za zbiranje podatkov.

Sledenje nam omogočajo osnovni podatki o paketih, ki so navedeni v *hdr_cmn* (angl. *common header*). Vsak paket ima svojo ID, tip paketa in velikost v bajtih (pove čas za dostavo paketa).

NS-2 pozna več različnih načinov zapisa za sledilne datoteke. *Old Trace Format*, *Wireless Trace Format* in *NAM trace format*. Najpogosteje se uporablja *Old Trace Format* znan tudi kot normalni zapis sledi. Za brezžična omrežja je na razpolago veliko različnih formatov, kot je najpogosteje uporabljeni *Old Wireless Trace Format*, *New Wireless Trace Format*, ki pomaga pri novejših orodjih za analizo sledilnih datotek in služi kot vhodna datoteka za to orodje ter ostali formati v brezžičnih omrežjih. Formati so namenjeni pri uporabi specifičnih omrežnih protokolih (AODV, DSDV, DSR, TORA). *NAM Trace Format* je vhodna datoteka za NAM, ki predstavlja vizualno predstavitev simulacije.

4.8.1 Nastavitve v simulacijski skripti

Na razpolago so naslednji ukazi[29], ki jih lahko vključimo v simulacijske skripte:

- `$ns_ trace-all $fd` - S tem ukazom omogočimo sledenje in zapis v datoteko `fd`.
- `$ns_ namtrace-all $namfd` - sledenje bo omogočeno za NAM in zapisano v datoteko `namfd`.

- `$ns_ namtrace-all-wireless $namfd X Y` - ukaz omogoči sledenje v brezžičnem omrežju in daje informacijo o velikosti topografije (x in y dimenziji).
- `$ns_ nam-end-wireless stoptime` - ta ukaz pove NAM, koliko časa (`stoptime`) je neaktiven.
- `$ns_ flush-trace` - ukaz sprosti medpomnilnik za zapisovanje sledilnih datotek in se navadno kliče takoj pred koncem izvajanja simulacije.
- `$ns_ use-newtrace` - ukaz izbere način zapisa *New Trace Format*
- `$ns_ use-taggedtrace` - ukaz izbere *Tagged Trace Format*

Nastavitve v `node-config` (glej Poglavje 4.5.3) zelo vplivajo na velikost in vsebino sledilnih datotek. Spodaj v Kodi 15 je prikazana vsebina simulacijske skripte, kjer so omogočena vsa sledenja, pred koncem simulacije pa sprostimo medpomnilnik za sledenje.

Koda 15 Vse sledenje je omogočeno.

```
set tracefd [open $opt(tr) w]
set namtrace [open $opt(nam) w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namtrace $opt(x) $opt(y)

$ns_ node-config -agentTrace ON \
-routerTrace ON \
-macTrace ON \
-movementTrace ON \

...

proc finish {} {
    global ns_ tracefd namfile
    $ns_ flush-trace
    close $tracefd
    close $namfile
    exit 0
}
$ns_ at $opt(stop) "finish"
```

4.8.2 Wireless Trace Format

V tem podpoglavju bomo opisali *Old Wireless Trace Format* znan tudi kot *NS2 Wireless Trace Format*. Dokumentacija za ostale načine izpisa je dosegljiva na [29].

```
s 21.500275000 _0_ MAC --- 0 AODV 106 [0 ffffffff 0 800] -----
[0:255 -1:255 30 0] [0x2 1 4 [1 0] [0 10]] (REQUEST)
```

Slika 4.7: Primer izpisa sledilne datoteke

Zapis na Sliki 4.7 lahko opišemo na naslednji način:

- Vozlišče `_0_` pošlje (prvi stolpec zapisa označuje vrsto dogodka `s`) paket ob času `21.500275000` sekund.
- Nivo sledi je na plasti `MAC`.
- Vozlišče ima številko ID `0`, protokol za usmerjanje, ki ga uporablja paket, je `AODV`. Paket ima velikost `106` bajtov.
- protokol `MAC` oceni, da je čas pošiljanja paketa po brezžičnem kanalu `0`.
- naslov `MAC` izvirnega vozlišča je `0`, naslov `MAC` cilja je `ffffffff`.
- Pošiljanje paketa poteka po omrežju Ethernet (`IP = 800`).
- naslov `IP` izvirnega vozlišča je `0`, `IP` naslov cilja je `1`.
- Številka vrat izvirnega vozlišča je `255`, številka vrat cilja pa prav tako `255`.
- `TTL` je `30` skokov, naslov naslednjega skoka je `0`.
- ID paketa je `0x2`.
- Število skokov je `1` in t.i. *broadcast ID* ima `4`.
- naslov `IP` cilja na tej poti je `1` in ima zaporedno številko `0`.
- naslov `IP` izvora na tej poti je `0` in ima zaporedno številko `10`.

- Zadnji stolpec (REQUEST) nam pove, da gre to za paket tipa RREQ (angl. *Route Request*).

Ostale možnosti, ki jih dovoljuje ta tip formata, so prikazane v Tabeli 4.3.

Tabela 4.3: Vsebina izpisa sledilne datoteke za *Old Wireless Trace Format* (~ns/trace/cmu-trace.cc).

Stolpec	Vsebina
1	Tip dogodka. r = prejeta (angl. <i>received</i>), s = poslano (angl. <i>sent</i> , f = posredovano (angl. <i>forwarded</i>), D = zavrženo (angl. <i>dropped</i>).
2	Čas ko se zgodi dogodek .
3	ID Vozlišča.
4	Tip sledilnega objekta. RTR = router, MAC = MAC plast, IFQ = <i>interface queue</i> , AGT = agent.
5	Razlog za dogodek. END = konec simulacije, COL = trk (angl. <i>collision</i>), DUP = podvojitev, ERR = napaka, RET = potek ponovnega štetja (angl. <i>retry count exceeded</i>), STA = napačno stanje, BSY = zaseden, DST = napaka ciljnega naslova, NRTE = ni poti do ciljnega vozlišča, LOOP = zanka poti, TTL = iztek časa TTL, IFQ = polna vrsta, TOUT = iztek časa paketa. CBK = povratni klic MAC, SAL = reševanje, ARP = zavržen zaradi ARP, FIL = filtriran, OUT = zavržen zaradi bazne postaje.
6	ID paketa.
7	Tip paketa.
8	Velikost paketa.
9-1	Pričakovani čas za pošiljanje podatkov.
9-2	naslov MAC cilja.
9-3	naslov MAC izvirnega vozlišča.
9-4	Tip (800 = IP, 806 = ARP)
10-1	naslov IP izvirnega vozlišča.
10-2	Številka vrat izvirnega vozlišča.
10-3	naslov IP cilja.
10-5	Številka vrat cilja.
10-6	Naslov za naslednji skok <code>next_hop_</code>
11 -	Specifično za vrsto paketa.

Poglavje 5

Namestitev NS-2

NS-2 simulator omrežja je sam po sebi zelo močno orodje za izvajanje simulacij. Največ težav nam povzročajo kvalitetni simulacijski scenariji, ki morajo biti čim boljši približek realnemu stanju.

Pri omrežju vozil smo si pomagali z generatorjem prometa *SUMO* (Simulation of Urban MObility).

5.1 SUMO

SUMO je odprtokodni simulator prometa. Ponuja bogato paleto funkcionalnosti kot so simulacija križišč, določanje hitrosti vozil, upoštevanje cestno prometnih pravil in uporaba podobnih primerov, ki jih srečamo v realnem svetu.

V NS-2 simulatorju nam SUMO pomaga zgenerirati premike vozil, ki služijo kot vhodna datoteka potrebna za izvedbo simulacije.

Z orodjem SUMO lahko zgeneriramo različne situacije, kot so različne gostote prometa v mestih, avtocestne situacije in podobne uporabne simulacije, ki jih lahko uporabim v omrežju vozil.

5.2 Open Street Map

OpenStreetMap je brezplačen projekt, ki ponuja zemljevid celega sveta. Zemljevid se lahko poljubno ureja in uporablja za nekomercialne namene. V NS-2 lahko del zemljevida (Primer: zemljevid Dunaja) uporabimo za topografijo našega simulacijskega scenarija. Ker nam je sam zemljevid brez vozil neuporabna, uporabimo orodje SUMO.

SUMO s pomočjo aplikacije *NETCONVERT* uvožen zemljevid iz OpenStreetMap pretvori v uporabno simulacijsko omrežje vozil. Nato pa s pomočjo *java* orodij, ki so del SUMO (*TraceExporter*) ali NS-2, generiramo naslednje datoteke:

Datoteka *Mobility*

```
$ns_ at 0.0 "$node_(642) setdest 598.35 290.15 1.83"
```

Slika 5.1: Primer izpisa mobility.tcl

Vsebina datoteke nam pove kam se bodo vozila premaknila v določenem trenutku. Primer na Sliki 5.1 pove, da se bo vozilo (vozlišče) z ID 640 v času 0.0 sekund in hitrostjo 1.83 m/s premaknilo na novo pozicijo z X koordinato 598.35 in Y koordinato 290.15.

CMU (omenjen v poglavju 4.7) je k NS-2 prispeval orodje *setdest*, ki generira veliko število vozlišč in definira njihovo premikanje.

Datoteka *Activity*

```
$ns_ at 0.0 "$ragent_(642) start"  
$ns_ at 282.0 "$ragent_(642) stop"
```

Slika 5.2: Primer izpisa activity.tcl

Datoteka *Activity* opisuje čas trajanja posameznega vozlišča. Na Sliki 5.2 je usmerjevalni agent 642 (agent je pripet na vozlišče) aktiven od časa 0.0

sekund do časa 282.0 sekund.

Datoteka *Config*

Koda 16 Vsebina konfiguracijske datoteke.

```
# set number of nodes
set opt(nn) 749.0

# set start/stop time
set opt(start) 0.0
set opt(stop) 599.0

# set grid size
set opt(x) 3002
set opt(y) 3002
set opt(min-x) -2
set opt(min-y) -2
```

Konfiguracijska datoteka vsebuje podatke o številu vozlišč, času trajanja celotne simulacije in velikosti zemljevida, ki ga bomo uporabili pri simulaciji.

5.3 Datoteka *Traffic-Connection*

V tem razdelku bomo opisali generiranje pomembne datoteke, ki omogoča pošiljanje prometa po omrežju, kar pa igra ključno vlogo pri simulacijah.

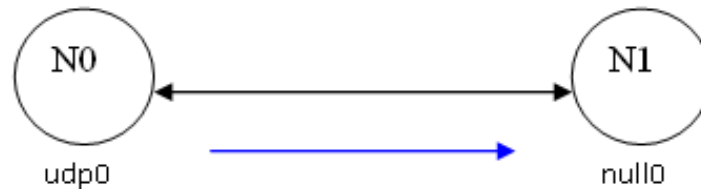
5.3.1 TCP/IP model v NS-2

Tabela 5.1: TCP/IP model

Aplikacijska plast	CBR, FTP, telnet
Transportna plast	TCP, UDP
Omrežna plast	Protokoli za usmerjanje
Povezovalna plast	Način prenosa: <i>simplex</i> , <i>duplex</i>
Fizična plast	MAC 802.11

- **Aplikacijsko plast** predstavljajo agenti kot so Telnet, FTP, CBR (angl. *Constant Bit Rate*).
- **Transportna plast** uporablja dva protokola. Hitrejši UDP (angl. *User Datagram Protocol*) in zanesljivejši TCP.
- **Omrežna plast** je odgovorna za usmerjevalne protokole in nizkonivojske protokole kot so ICMP, IP in IGMP.
- **Povezovalna plast** zajema dejanski prenos podatkov in preslikavo naslovov IP v naslove MAC. Povezovalna plast omogoča simplex in duplex način prenosa podatkov.
- **Fizična plast** je najnižja plast modela in je direktno priključena na brezžični kanal.

5.3.2 Primer UDP komunikacije v NS-2



Slika 5.3: Komunikacija poteka od UDP Agenta do Null Agenta

Koda 17 UDP Agenta priprnemo na vozlišče N0.

```
#Create a UDP agent and attach it to node n0
```

```
set udp0 [new Agent/UDP]
```

```
$ns attach-agent $n0 $udp0
```

```
# create a null agent which act as traffic sink
```

```
# and attach it to node n1
```

```
set null0 [new Agent/Null]
```

```
$ns attach-agent $n1 $null0
```

```
# connect two agents with each other
```

```
$ns connect $udp0 $null0
```

5.3.3 Generator prometa CBR

Za sodelovanje različnih programov na oddaljeni in sprejemni strani, potrebujemo na aplikacijski plasti protokol, ki bo omogočal generiranje prometa. CBR (angl. constant bit-rate) je največkrat uporabljen pri izvajanju simulacij. Lahko pa uporabimo tudi agente FTP, Telnet in podobne.

Koda 18 Primer uporabe CBR.

```
# creating CBR agent
set cbr0 [new Application/Traffic/CBR]

# Attach the CBR agent to some udp/tcp agent
$cbr0 attach-agent $udp0
```

Koda 19 Življenjski cikel CBR agenta.

```
$ns at 1.0 \"$cbr0 start"
$ns at 4.0 \"$cbr0 stop"
```

NS-2 vsebuje svojo skripto za generiranje prometa imenovano **cbrgen**. Skripta se nahaja v mapi `~ns/indep-utils/cmu-scen-gen`. Omogoča generiranje CBR ali TCP prometa med brezžičnimi vozlišči. Primer ukazne vrstice za *cbrgen* orodje je sledeč:

```
ns cbrgen.tcl -type cbr -nn 10 -seed 1.0 -mc 8 -rate 4.0 >cbr-10.tcl
```

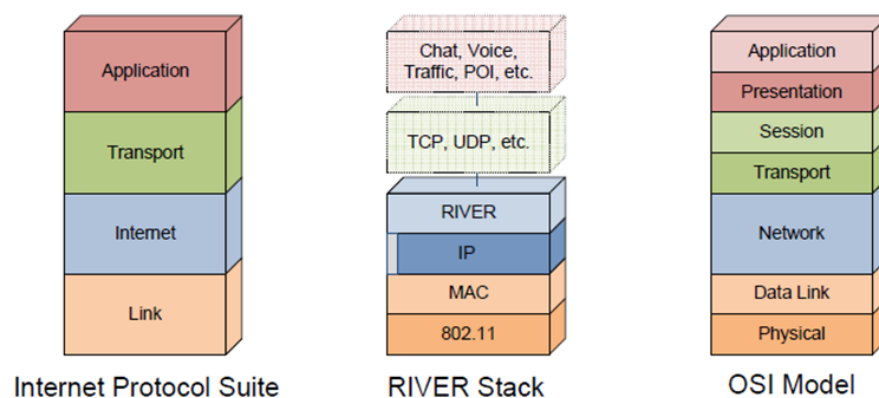
CBR povezava med desetimi vozlišči, kjer imamo maksimalno 10 povezav.

Poglavje 6

Implementacija protokola RIVER

6.1 OSI model za RIVER

Protokol RIVER uporablja standard 802.11 za brezžično komunikacijo med vozili. Povezovalna plast predstavlja protokol MAC. Del omrežne plasti pa sestavlja protokol RIVER in IP naslavljanje. Pri tipičnem IP naslavljanju se uporablja IP naslov za ciljno vozlišče. V primeru uporabe protokola RIVER pa je IP naslov določen z fizično lokacijo vozlišča.



Slika 6.1: Plasti ISO OSI model za protokol RIVER

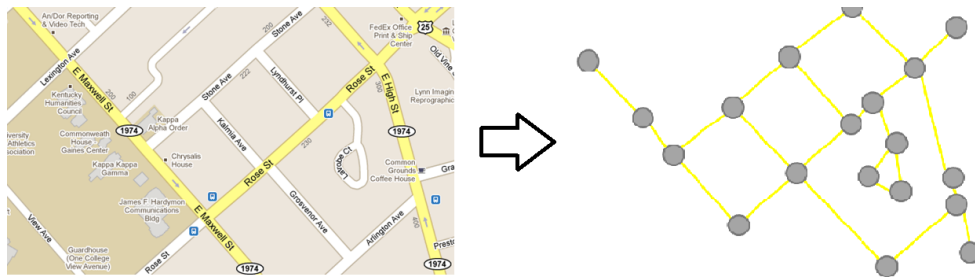
6.2 Beacon

Vsako vozlišče oddaja sporočila v uporabniško določenem intervalu. Ta pozdravna sporočila se ne posredujejo naprej do ostalih vozlišč, ampak samo do sosednjih vozlišč in so namenjena obveščanju sosednjih vozlišč o svoji lokaciji. Pri implementaciji pozdravna sporočila vsebujejo IP naslov pošiljatelja, fizično lokacijo vozlišča in čas, ko je bilo sporočilo poslano. Naše pozdravno sporočilo vsebuje še informacijo o zanesljivosti pošiljanja paketov preko tega vozlišča. Vsako vozlišče vsebuje tabelo sosednih vozlišč, ki jo osveži ob prejemu pozdravnega sporočila. Protokol RIVER pošilja tudi implicitna pozdravna sporočila preko sporočil, kot so sonar sporočila (angl. *piggybacking*).

6.3 Cestni graf

RIVER za usmerjanje paketov po omrežju uporablja neusmerjen graf, katerega vozlišča so cestna križišča ali cestni zavoji. Povezave v grafu so ceste, ki povezujejo križišča ali cestne zavoje. Tako usmerjanje poteka samo po cestah, kjer se vozlišča (vozila) dejansko gibljejo. Preslikava zemljevida v graf je prikazana na Sliki 6.2

Graf je določen s parom $G = (V, E)$, kjer je V množica vozlišč, E pa množica povezav med vozlišči. Posamezno vozlišče je določeno ID številko, ki je unikatna za celoten graf in fizično lokacijo vozlišča. Povezava med vozlišči vsebuje informacije o dveh vozliščih med katerima se nahaja ta povezava, ter podatek o uteži w za povezavo, ki se uporablja pri računanju poti do ciljnega vozlišča.



Slika 6.2: Preslikava zemljevida v neusmerjen graf

6.4 Komponenta nadzora prometa

6.4.1 Sonar sporočila

Zaradi omejitev in realnega stanja v cestnem prometu (dela na cesti, slabo vreme, cestno prometni predpisi) pogosto prihaja do lukenj v omrežju, kjer se paketov ne da posredovati.

Protokol RIVER pošilja sonar sporočila med vozili, ki so blizu vozlišč cestnega grafa, vzdolž povezave med dvema vozliščema. Sonar sporočilo je sestavljeno iz dveh faz pošiljanja. Prva faza je pošiljanje sonar sporočila vzdolž povezave do vozila, ki je blizu sosednjega vozlišča. Druga faza pa vsebuje pošiljanje potrditvenega sporočila nazaj do pošiljatelja sonar sporočila. S sklepnim dejanjem pošiljanja je ugotovljeno, da na povezavi (cestnem odseku) ni lukenj in so podatkovni paketi lahko posredovani preko te povezave.

Sonar sporočila ¹ se pošiljajo glede na določen časovni interval, ali glede na pretekli čas od zadnjega testiranja zanesljivosti povezave. Sonar sporočilo se pošilja nedoločenemu prejemniku. Znana je samo množica začetnih prejemnikov, ki se nahajajo na določeni fizični lokaciji. Če sonar sporočilo naleti na luknjo v omrežju, je zavrženo.

¹Avtor Bernsen jih poimenuje *Probes*. Poimenovanje *sonar sporočila* je zaradi podobnosti delovanja sonar naprave.

6.4.2 Lista povezav - KEL

Vsak paket protokola RIVER, ki se pošilja po omrežju, vsebuje podatkovno strukturo KEL (angl. *Known Edge List*), ki je lista znanih povezav in pripadajočih uteži. Utež je določena glede na nedavno razpoložljivost povezave za pošiljanje paketov po njej. Povezave, kjer so sonar sporočila zavržena, dobijo neskončno vrednost za utež. Posamezna povezava in njena utež sta veljavni določen čas. Po preteku tega časa si posamezen paket lahko osveži listo povezav in s tem določi novo zanesljivost posameznih povezav.

6.5 Potek iskanja poti

Pri iskanju poti si protokol RIVER pomaga z algoritmom Dijkstra. Na podlagi dane uteži w za povezavo, določi najugodnejšo pot do ciljnega vozlišča.

Podobno kot pri protokolu CAR, se pri določanju poti do ciljnega vozlišča, določijo tudi sidriščne točke, ki se hranijo v glavi poslanega sporočila. Če je ciljno vozilo v bližini izvirnega vozila, poteka pošiljanje sporočila na način požrešnega posredovanja.

V primeru zavrnitve na povezovalni plasti, se postopek pošiljanja ponovi. Pri vsakem skoku paketa iz vozila na vozilo se zgodi sledeče:

- vozlišče, ki je prejelo paket, preveri za obstoj ciljnega vozlišča v bližini,
- posredno si osveži listo sosednjih vozlišč,
- osveži si vrednosti uteži na povezavah glede na KEL in izračunano pot do ciljnega vozlišča,
- posreduje paket do naslednje sidriščne točke, oziroma požrešno posreduje paket do ciljnega vozlišča.

Paketi se pri večini znanih usmerjevalnih protokolov včasih znajdejo v brezizhodnem položaju in morajo biti zavrženi. Protokol RIVER, pa v brezizhodnem položaju ponovno uporabi algoritem Dijkstra in vrne paket do zadnje sidriščne točke in od tam določi pot do ciljnega vozlišča.

Protokol RIVER je optimiziran tako, da paket zavrne čim kasneje. Pomembna optimizacija je ta, da v sidriščnih točkah predvideva premike vozil in tako posreduje paket vozilu, ne glede na to da še ni v območju sidriščne točke.

Poglavje 7

Izvajanje simulacije z NS-2

Izvedba simulacije v NS-2 se začne z izgradnjo OTcl simulacijske skripte. V simulacijsko skripto vključimo komponente (gradnike) in opredelimo dogodke, ki se bodo zgodili (Primer: premik vozlišča). Preprost primer simulacijske skripte bi bil sledeč: topografija omrežja vsebuje dva vozlišča, med njima je vzpostavljena duplex povezava s pasovno širino 5Mb/s, po povezavi poteka prenos datotek (FTP) preko protokola TCP. Vse to se začne in konča v uporabniško določenem času.

7.1 Glavni gradniki

Simulacijski scenarij je zgrajen iz treh gradnikov:

- Topografija omrežja (vozila, vozlišča)
- Povezave, promet, protokoli (agenti)
- Dogodki in napake

Pod *topografijo omrežja* so mišljena vozlišča in povezave med vozlišči. To lahko naredimo ročno, ali pa s pomočjo generatorjev opisanih v razdelku 5.2. Povezave in promet po omrežju lahko tudi vzpostavimo ročno ali pa zgeneriramo s pomočjo orodij. Pod dogodke in napake spada pretok paketov, izguba paketov, premik mobilnih vozlišč, vzpostavitev in prekinitve povezave.

7.2 Simulacijski scenarij

Scenarij, potreben za izvedbo simulacije, je sledeč:

1. Ustvarimo instanco simulatorja
2. Odpremo sledilne datoteke (*Wireless Trace* in *NAM Trace*)
3. Konfiguriramo parametre za vozlišča
4. Ustvarimo vozlišča in povezave med njimi
5. Ustvarimo agente in aplikacije (protokole - CBR, UDP)
6. Določimo čas trajanja prometa in agentov
7. Dodamo proceduro `finish()`
8. Kličemo metodo za vstavitev na koncu simulacije
9. Zaženemo simulacijo z ukazom `run`

7.3 Zagon simulacije

Če želimo zagnati simulacijo, moramo najprej imeti sestavljeno simulacijsko skripto. Primer zagona simulacije bomo izvedli za zemljevid velikosti 3000 x 3000 (generiran s pomočjo orodja SUMO), 749 vozil, trajanje simulacije 599 sekund, uporabo usmerjevalnega protokola AODV in aplikacijskega prometa CBR preko komunikacije UDP. Po korakih iz razdelka 7.2 naredimo naslednje postopke v skripti *wireless-aodv-p2.tcl*:

Instanca simulatorja

Koda 20 Ustvarimo instanco simulatorja.

```
set ns_ [new Simulator]
```

Sledilne datoteke

Koda 21 Ustvarimo instanco simulatorja.

```
set tracefd [open $opt(tr) w]
%opt(tr) destination file for trace
set namfile [open $opt(nam) w]
%opt(nam) destination file for nam trace
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless $namfile $opt(x) $opt(y)
```

Vozlišča in povezave med njimi

Koda 22 Ustvarimo mobilno vozlišče z parametri za AODV.

```
for {set i 0} {$i < $opt(nn)} {incr i} {
%opt(nn) number of all nodes
    aodv-create-mobile-node $i
}
```

Za aplikacijske agente in povezave med vozlišči v skripto uvozimo *Traffic-Connection* datoteko z vrstico `source grid_11x300/grid_11x300.con_p2.tcl`

Prav tako uvozimo vse tri datoteke, *Mobility*, *Activity* in *Configuration*, ki smo jih zgenerirali s pomočjo SUMO orodja.

finish() metoda

Koda 23 Finish metoda ki jo kličemo na koncu skripte

```
proc finish {} {
    global ns_ tracefd namfile
    $ns_ flush-trace
    close $tracefd
    close $namfile
    exit 0
}
```

Na koncu ne smemo pozabiti na zagon simulacije v vrstici `$ns_ run`.

Z naslednjim ukazom v terminalu operacijskega sistema Linux zaženemo simulacijsko skripto in dobljene rezultate zapišemo v datoteko, katero bomo kasneje analizirali:

```
:~$ ns wireless-aodv-p2.tcl
```

Po zaključku simulacije dobimo dve izhodni datoteki *aodv_file.tr* in *aodv_file.nam*. Prvo datoteko bomo uporabili za analizo dobljenih podatkov, drugo datoteko pa lahko uporabimo za vizualno predstavitev poteka pošiljanja podatkov po mobilnem omrežju z uporabo orodja NAM.

Poglavje 8

Analiza dobljenih rezultatov

Po uspešno končani simulaciji, lahko dobljeno sledilno datoteko analiziramo. Odvisno od velikosti simulacije in predmeta analize, lahko za analizo uporabimo različna orodja, kot so *Trace graph* [31], NS-2 Wireless Trace Analyzer, ali pa sami spišemo skripte s pomočjo programskega jezika Perl, Python in podobnih. V primeru ogromne količine podatkov, so pri analizi sledilnih datotek uporabne skripte. V tem poglavju bomo omenili uporabo skripte AWK. Za grafično predstavitev pa nam je na razpolago GNUPlot, Xgraph in veliko drugih programov za izris grafov.

8.1 Simulacijske meritve

- Razmerje med poslanimi in prejetimi paketi (angl. *Packet Delivery Ratio*)
- Razmerje nepotrebnih računanj poti (angl. *Routing Overhead*)
- Časovne izgube (angl. *End-to-end Delay*)
- Izguba podatkov (angl. *Data packet loss*)
- Količina poslanih podatkov na sekundo (angl. *Throughput*)

Zgoraj je nekaj naštetih dogodkov, ki se dogajajo pri simulaciji in jih lahko uporabimo npr. za primerjavo zmogljivosti različnih usmerjevalnih protokolov v omrežju vozil.

8.2 AWK skripta

Koda 24 Primer uporabe skripte za *Packet Delivery Ratio*.

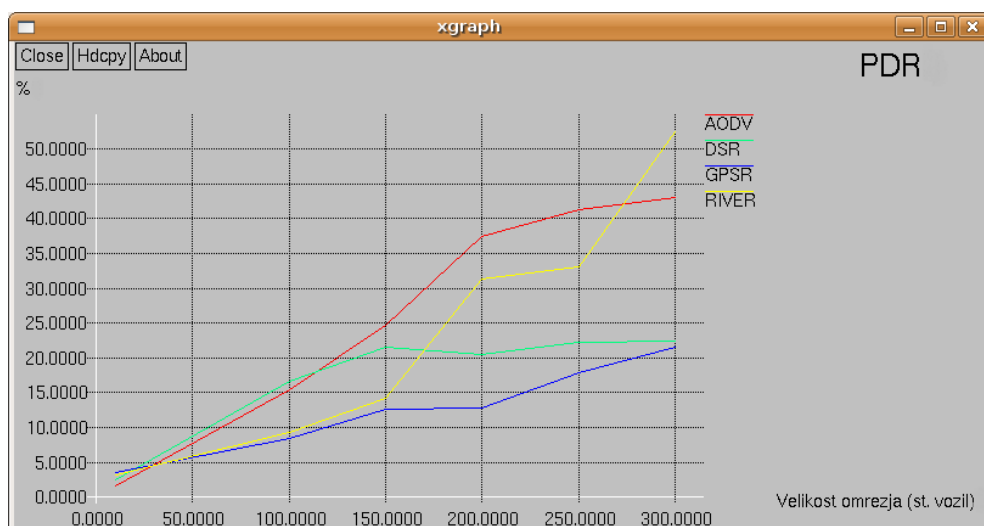
```
BEGIN {  
    sendLine = 0;  
    recvLine = 0;  
    fowardLine = 0;  
}  
  
$0 ~/^s.* AGT/ {  
    sendLine ++ ;  
}  
  
$0 ~/^r.* AGT/ {  
    recvLine ++ ;  
}  
  
$0 ~/^f.* RTR/ {  
    fowardLine ++ ;  
}  
  
END {  
    printf "cbr s:%d r:%d, r/s Ratio:%.4f, f:%d \n",  
        sendLine, recvLine, (recvLine/sendLine), fowardLine;  
}
```

8.3 Rezultati primerjav

Simulacije smo izvajali s pomočjo NS-2 simulatorja verzije 2.34. Protokol RIVER semo primerjali s protokoli AODV, DSR in GPSR. Na transportni plasti smo generirali CBR promet med petimi naključno izbranimi vozili. Uporaba prometa CBR predstavlja pošiljanje zvoka in slike preko omrežja. Vsako meritev smo ponovili večkrat, da smo lahko določili povprečje meritve. Za primerjavo protokolov smo določili pet meril (Razmerje med poslani in prejetimi paketi, Število nepotrebno generiranih poti za usmerjanje, Količina poslanih podatkov na sekundo, Število posredovanj podatkovnih paketov, Časovne izgube, Izguba podatkov).

Za potrebe simulacije smo s pomočjo orodja SUMO generirali zemljevid *manhattanske* oblike, velikosti $2460m \times 2460m$. Populacija vozil na zemljevidu je bila med 10 in 300 vozili. Vsaka simulacija je uporabljala enake datoteke *Mobility*, *Activity* in *Config* (Datoteke je generiralo orodje SUMO opisano v 5.2).

8.3.1 Razmerje med poslani in prejetimi paketi

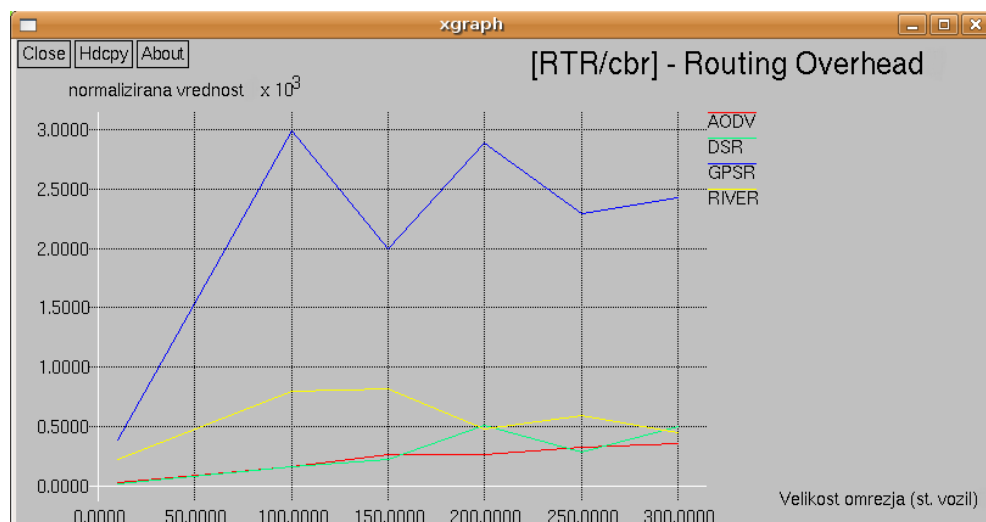


Slika 8.1: angl. *Packet Delivery Ratio* v odvisnosti od števila vozil

Pošiljanje podatkovnih paketov je potekalo med petimi pošiljatelji/prejemniki z uporabo CBR in prenosom podatkov 512 kb/s. Kar pomeni, da je vsako vozilo generiralo podatkovni paket na vsake 8 sekund. Vsak je poslal 21 paketov, kar skupno nanese 105 podatkovnih paketov. Simulacija je tekla 200 sekund.

Iz grafa na Sliki 8.1 je razvidno, da uspešnost pošiljanja podatkovnih paketov pri protokolu RIVER z večanjem števila vozil, narašča. Najslabše se za to meritev izkaže, kljub svoji popularnosti, protokol GPSR. Protokol RIVER je med izbranimi protokoli najuspešnejši pri dostavi podatkovnih paketov do ciljnih vozil. Protokol AODV je najboljši za manjšo populacijo vozil. Za protokol GPSR lahko predvidevamo, da bi se z večanjem gostote vozil izboljšal pri dostavi podatkovnih paketov.

8.3.2 Število nepotrebno generiranih poti za usmerjanje



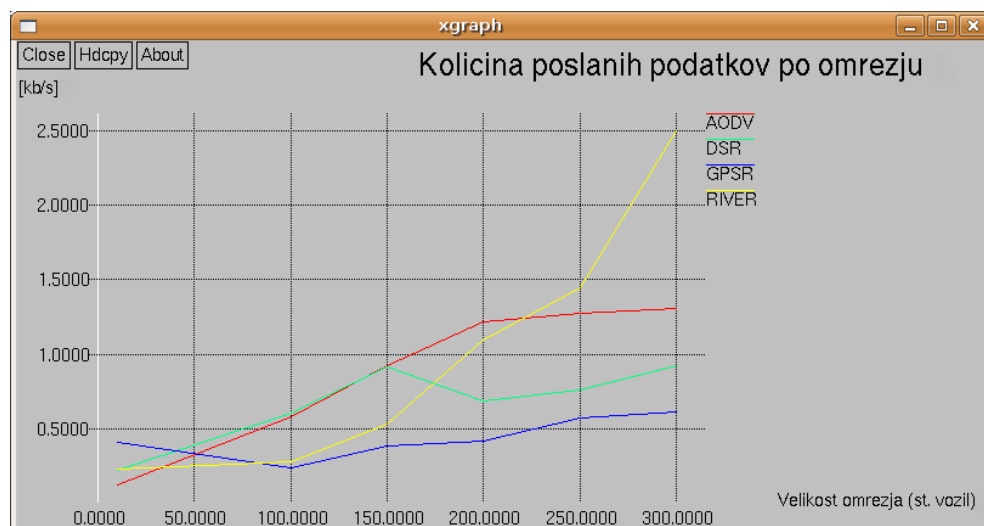
Slika 8.2: angl. *Routing Overhead* v odvisnosti od števila vozil

MANET protokola (AODV in DSR) prilagojena za uporabo v omrežju vozil, nimata veliko preračunavanj poti. Svoje poti računata le na zahtevo in še to

s pomočjo pozdravnih sporočil. Protokol GPSR za vsako luknjo v omrežju, ponovno preračunava pot do ciljnega vozlišča. Krivulja protokola RIVER na grafu kaže zelo obetajoče rezultate. V Razdelku 6.5 smo omenili, da RIVER poskuša čim kasneje zavrniti paket. Iz grafa na Sliki 8.2 je lepo razvidna ta prednost usmerjevalnega protokola.

Za to metriko so si protokoli RIVER, AODV in DSR konkurenčni. Ker želimo, da se po omrežju ne pošilja prevelika količina nepotrebnih podatkov (vpliva tudi na zasedenost prenosnega medija), je protokol GPSR najslabši za to meritev.

8.3.3 Količina poslanih podatkov na sekundo



Slika 8.3: angl. *Data Packet Throughput* v odvisnosti od števila vozil

Zgornji graf kaže na veliko količino prenesenih podatkov na sekundo za protokol RIVER. To pomeni večjo natančnost pri določanju in računanju poti do ciljnih vozlišč. Torej se protokol GPSR za to metriko slabo izkaže. Stagniran naklon krivulj protokolov GPSR, DSR in AODV kaže na to, da tudi z večanjem gostote vozil, ne bi izboljšali količine poslanih podatkov na sekundo. Tako je najboljši za to metriko protokol RIVER, ki tudi zelo izstopa

pri večji gostoti vozil.

8.3.4 Število posredovanj podatkovnih paketov



Slika 8.4: angl. *Packet Forward* v odvisnosti od števila vozil

Protokol RIVER je pri tej meritvi nekje v povprečju. Protokol DSR zelo dobro posreduje pakete pri večjem številu vozil. Kar pa ne pomeni, da je boljši za to metriko. Protokola AODV in GPSR sta najboljša pri tej metriki, saj opravita malo posredovanj za naš scenarij. Kako uspešna sta bila pri tem posredovanju, pa smo že predstavili v prvi metriki. V naslednjem podrazdelku bomo videli, da je večje število posredovanj podatkovnih paketov vplivalo na časovne izgube pri pošiljanju.

8.3.5 Časovne izgube



Slika 8.5: angl. *End to End Delay* v odvisnosti od števila vozil

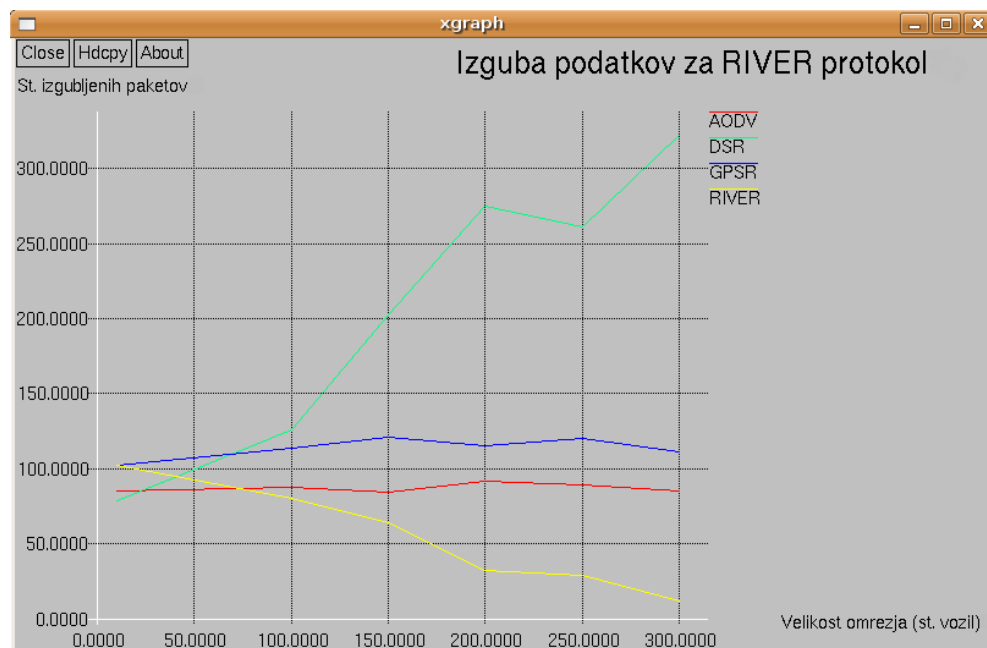
Graf prikazuje koliko časa v povprečju potrebujejo paketi, da prepotujejo pot od pošiljatelja do prejemnika. Naše simulacije so pokazale, da ima protokol GPSR najmanjše časovne izgube. Protokol RIVER je v tem primeru ugodnejši za manjšo populacijo vozil, z večanjem gostote vozil, se mu časovne izgube večajo. To pa ni dobro za to metriko.

Zgoraj omenjene meritve so pokazale, da ima protokol RIVER nekaj prednosti in nekaj slabosti. Njegova slabost je veliko število posredovanih paketov. Vendar ga ta slabost privede do višjega razmerja pri dostavljanju paketov. Nekatere meritve so lahko malo zavajajoče. V mislih imamo metriko uspešnosti pri dostavljanju paketov. Protokol GPSR je se za naš scenarij res slabo izkazal. Vendar moramo poudariti, da smo simulacije časovno omejili. To pa pomeni, da veliko paketov ostane v omrežju in bi s povečanjem časa izvajanja simulacije, povečali tudi performanco nekaterih protokolov.

Za protokol GPSR to zagotovo [11] velja.

Naslednja metrika nam bo pomagala priti do končnih sklepnih ugotovitev.

8.3.6 Izguba podatkovnih paketov



Slika 8.6: angl. *Data Packet Loss* v odvisnosti od števila vozil

Protokol DSR se za to metriko najslabše izkaže. Protokola AODV in GPSR pa imata zadovoljive rezultate. Protokol RIVER ima pri tej metriki tako dobre rezultate za gosto poseljen zemljevid z vozili zato, ker smo pri implementaciji protokola RIVER omenili, da se protokol s svojimi algoritmi trudi poslati paket do ciljnega vozila. Zato so paketi čim kasneje zavrženi in ima protokol malo število izgub.

Poglavje 9

Zaključek

V diplomskem delu smo predstavili različne usmerjevalne algoritme. Opredelili smo njihove prednosti in slabosti. Podrobno smo pregledali arhitekturo omrežnega simulatorja NS-2, ter predstavili namestitvev simulatorja. Pri namestitvi omrežnega simulatorja smo naleteli na številne težave. Verzijo 2.34 smo si izbrali, ker je preizkušena in je pri namestitvi simulatorja in njegovih komponent povzročala najmanj težav. Predstavili smo usmerjevalni protokol RIVER in ga implementirali v NS-2. Navedli smo postopke za izvajanje simulacij v simulatorju in izvedli nekaj simulacij, za potrebe performančnih primerjav z drugimi usmerjevalnimi algoritmi. Nato smo primerjali protokol RIVER s protokoli AODV, DSR in GPSR. Rezultati primerjav so pokazali, da ima RIVER zaradi svoje komponente nadzora prometa velik potencial pri nadaljnjem razvijanju tega protokola.

Primerjani testi tudi kažejo, da se algoritmi pri nizki gostoti prometa slabo izkažejo. Zato menimo, da se bodo usmerjevalni algoritmi prilagajali potrebam na terenu. Torej za potrebe mestnega prometa bo uporabljena drugačna strategija, kakor tam kjer je vozil zelo malo, ali pa so ločena in tvorijo več posamičnih, ločenih omrežij vozil. Protokol, ki se bo dobro znašel v taki situaciji, bo zagotovo naredil nov mejnik v razvoju usmerjevalnih omrežnih protokolov.

Če pogledamo kakšno je stanje uporabe usmerjevalnih algoritmov v real-

nem svetu, lahko ugotovimo, da avtomobilski giganti (BMW, GM, Nissan, Toyota, Ford, Audi, DaimlerChrysler) usmerjajo svoj razvoj na področje omrežja vozil. Omejuje jih nedostopnost obcestnih infrastruktur, pomanjkanje koordinacije med avtomobilskimi podjetji, problem varnosti pošiljanja sporočil ter zasebnost vozil. Tako lahko samo upamo, da bomo videli razvoj uporabe omrežja vozil tudi v realnosti.

Slike

2.1	Usmerjevalni protokoli za omrežja vozil	7
2.2	Tehniki <i>Greedy forwarding</i> in <i>Face routing</i>	14
4.1	Povezljivost programskih jezikov OTcl in C++	31
4.2	6 kategorij razredne hierarhije v NS-2	32
4.3	Paketni format v NS-2	34
4.4	Ožičeno vozlišče (<i>unicast</i> in <i>multicast</i>)	37
4.5	Shema mobilnega vozlišča v NS-2	38
4.6	Primer preproste topologije v NS-2	44
4.7	Primer izpisa sledilne datoteke	54
5.1	Primer izpisa mobility.tcl	58
5.2	Primer izpisa activity.tcl	58
5.3	Komunikacija poteka od UDP Agenta do Null Agenta	61
6.1	Plasti ISO OSI model za protokol RIVER	63
6.2	Preslikava zemljevida v neusmerjen graf	65
8.1	Razmerje med poslanimi in prejetimi paketi v odvisnosti od števila vozil	75
8.2	Graf nepotrebno generiranih poti za usmerjanje	76
8.3	Količina poslanih paketov v odvisnosti od števila vozil	77
8.4	Graf paketnih posredovanj v odvisnosti od števila vozil	78
8.5	Časovne izgube v odvisnosti od števila vozil	79
8.6	Izguba podatkovnih paketov v odvisnosti od števila vozil	80

Literatura

- [1] N. Abramsson, "The ALOHA system - another alternative for computer communications," *AFIPS Conf. Proc.*, št. 37, str. 695–702, 1970.
- [2] Continuous Air Interface for Long and Medium Interface. Dostopno na: <http://www.calm.hu>.
- [3] The Vehicle Infrastructure Integration Initiative. Dostopno na: <http://www.vehicle-infrastructure.org>.
- [4] DSRC/WAVE Prototype. Dostopno na: http://www.unwired.ee.ucla.edu/dsrc/dsrc_testbed_simple.htm.
- [5] (2009) Research and Prototype Experience on WAVE Technology. Dostopno na: http://cms.comsoc.org/SiteGen/Uploads/Public/Docs_Globecom_2009/-4-WAVE_Talk_Globecom_2009.pdf.
- [6] Kevin C. Lee, Uichin Lee in Mario Gerla, "Survey of Routing Protocols in Vehicular Ad Hoc Networks," *Advances in Vehicular Ad-Hoc Networks: Developments and Challenges*, str. 153.
- [7] A. Iwata, "Scalable Routing Strategies for Ad-hoc Wireless Networks," *IEEE JSAC*, str. 1369–1379, 1999.
- [8] D. B. Johnson in D. A. Maltz, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing, T. Imielinski in H. Korth*, Poglavje 5, str. 153–181.

-
- [9] V. D. Park in M. S. Corson, "A highly adaptive distributed routing algorithm for mobile wireless networks," INFOCOM '97. Sixteenth Annual Joints Conference of the IEEE Computer and Communications Societies. Proceedings IEEE, št. 3, str. 1405-1413, št. 3 str. 7-12, 1997.
 - [10] C. Perkins, E. Belding-Royer in S. Das "Ad hoc On-Demand Distance Vector (AODV) Routing," julij 2003.
 - [11] B. Karp, H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks.", v *Mobile Computing and Networking*, str. 243–254, 2000.
 - [12] H. Fußler, M. Mauve, H. Kasemann, D. Vollmer, "Location-Based Routing for Vehicular Ad Hoc Networks," *Mobile Computing and Communication Review*, št. 1, november 2002.
 - [13] C. Lochert, M. Mauve, H. Fussler, H. Hartenstein, "Geographic routing in city scenarios," *SIGMOBILE Mob. Comput. Commun.*, št. 9, zv. 1, str. 69–72, 2005.
 - [14] V. Naumov, T. R. Gross, "Connectivity-Aware Routing (CAR) in Vehicular Ad-hoc Networks," *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, str. 1919–1927, maj 2007.
 - [15] C. Lochert, H. Hartenstein, J. Tian, H. Fussler, "A routing strategy for vehicular ad hoc networks in city environments," *Intelligent Vehicles Symposium*, str. 156–161, junij 2003.
 - [16] Dijkstra's Algorithm. Dostopno na:
http://en.wikipedia.org/wiki/Dijkstra's_algorithm.
 - [17] B. Seet, G. Liu, B. Lee, "A-STAR: A Mobile Ad Hoc Routing Strategy for Metropolis Vehicular Communications," *NETWORKING 2004*, str. 989–999.

-
- [18] D. Forderer, "Street-Topology Based Routing," *Doktorska disertacija, University of Mannheim*, maj 2005.
- [19] M. Jerbi, S. Senouci, R. Meraihi, "An improved vehicular ad hoc routing protocol for city environments," *Communications, 2007. ICC '07. IEEE International Conference*, str. 3972–3979, junij 2007.
- [20] K. Lee, M. Le, J. Haerri in M. Gerla, "Louvre: Landmark overlays for urban vehicular routing environments," *Proceedings of IEEE WiVeC*, 2008.
- [21] H. Fussler, H. Hannes, W. Jorg, M. Martin, E. Wolfgang, "Contention-Based Forwarding for Street Scenarios," *Proceedings of the 1st International Workshop in Intelligent Transportation*, Hamburg, Germany, str. 155–160, marec 2004.
- [22] K. C. Lee, U. Lee, M. Gerla, "TO-GO: TOpology-assist geo-opportunistic routing in urban vehicular grids," *Wireless On-Demand Network System and Services. WONS 2009*, str. 11–18, februar 2009.
- [23] J. Zhao, G. Cao, "VADD: Vehicular-Assisted Data Delivery in Vehicular Ad Hoc Networks," *INFOCOM 2006. 25th IEEE*, str. 1–12, april 2006.
- [24] I. Leontiadis, C. Mascolo, "GeoOpps: Geographical Opportunistic Routing for Vehicular Networks," *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007*, str. 1–6, junij 2007.
- [25] P. C. Cheng, J. T. Weng, L. C. Tung, K. C. Lee, M. Gerla in J. Harri, "GeoDTN+NAV: A Hybrid Geographic and DTN Routing with Navigation Assistance in Urban Vehicular Networks," *Proceedings of the 1st International Symposium on Vehicular Computing Systems (ISVCS'08)*, Dublin, Ireland, julij 2008.
- [26] J. Bernsen, "A reliability-based routing protocol for vehicular ad-hoc networks," *University of Kentucky Master's Theses*. Dostopno na: http://uknowledge.uky.edu/gradschool_theses/132.

-
- [27] (2008) *A Survey of Network Simulation Tools: Current Status and Future Developments*. Dostopno na:
<http://www.cse.wustl.edu/~jain/cse567-08/ftp/simtools/>.
- [28] *The VINT (Virtual InterNetwork Testbed) Project homepage*. Dostopno na: <http://www.isi.edu/nsnam/vint/index.html>.
- [29] K. Fall in K. Varadhan. *The ns Manual (formerly ns Notes and Documentation)*, November 2011. Dostopno na:
<http://www.isi.edu/nsnam/ns/doc/index.html>.
- [30] David B. Johnson in David A. Maltz, *Protocols for Adaptive Wireless and Mobile Networking*, IEEE Personal Communications, februar 1996.
- [31] *Trace graph homepage*. Dostopno na:
<http://www.geocities.com/tracegraph>.
- [32] *NS2Ultimate homepage*. Dostopno na:
<http://www.ns2ultimate.com>.
- [33] L. Lamport. *LaTEX: A Document Preparation System*. Addison-Wesley, 1986.