UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

LUKA BIRSA

ODLOŽLJIVA OMREŽJA IN PROPHET USMERJEVALNI PROTOKOL

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

Ljubljana, 2009

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01551/2009
Datum: 15.03.2009

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

Kandidat: **LUKA BIRSA**

Naslov: **ODLOŽLJIVA OMREŽJA IN PROPHET USMERJEVALNI PROTOKOL**
DELAY TOLERANT NETWORKS AND PROPHET ROUTING PROTOCOL

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Analizirajte razloge za nastanek odložljivih računalniških omrežij in podajte pregled obstoječih rešitev. Opišite nastanek usmerjevalnega protokola Prophet in podrobno razložite njegove lastnosti. Izberite tehnologijo s pomočjo katere je mogoče prikazati delovanje odložljivega omrežja in protokola Prophet ter oceniti njegovo uporabnost v praksi. Na izbrani tehnologiji izdelajte programsko opremo in preizkusite delovanje tega protokola. Ocenite njegovo delovanje in morebitne možnosti za izboljšave.

Mentor:
prof. dr. Dušan Kodek

Fakulteta za računalništvo in informatiko
2
Univerza v Ljubljani

Dekan:
prof. dr. Franc Solina

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

LUKA BIRSA

ODLOŽLJIVA OMREŽJA IN PROPHET USMERJEVALNI PROTOKOL

DIPLOMSKO DELO NA UNIVERZITETNEM ŠTUDIJU

MENTOR: PROF. DR. DUŠAN KODEK

Ljubljana, 2009

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a **Luka Birsa**,

z vpisno številko **63000139**,

sem avtor/-ica diplomskega dela z naslovom:

Odložljiva omrežja in PRoPHET usmerjevalni protokol

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom (naziv, ime in priimek) **prof. dr. Dušana Kodeka**
- 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 14.9.2009 Podpis avtorja/-ice: _____

Zahvala

Za vso strokovno pomoč, podporo in predvsem potrpljenje pri mojem dolgotrajnem postopku priprave diplomskega dela se iskreno zahvaljujem svojemu mentorju prof. dr. Dušan Kodeku.

Za pomoč pri pripravi implementacije se zahvaljujem prof. Kaustubhu S. Phanseju in dr. Andersu Lindgrenu. Zahvaljujem se tudi Christophu Baraerju in Oskarju Burmanu za pomoč pri pripravi rešitev in vso delo, ki sta ga opravila v sklopu Sami Network Connectivity projekta.

Zahvaljujem tudi svoji družini - Olgi, Urošu in Jaki, moji punci Manci in prijatelju Urošu, ki so me podpirali med vsemi študijskimi aktivnostmi, katerih rezultat je pričujoče delo.

Kazalo

Povzetek

Zaradi vsesplošne razširjenosti kakovostne omrežne povezljivosti v razvitem svetu danes hitro spregledamo potrebo po omrežjih, ki bi omogočala komuniciranje tudi če infrastrukture ni na voljo. Omrežja, ki omogočajo komunikacijo v takšnih okoljih, imenujemo odložljiva omrežja in za svoje delovanje ne potrebujejo stalne povezave med vozlišči, ki (si) izmenjujejo podatke.

V tem diplomskem delu je opisan postopek nastanka implementacije PRoPHET usmerjevalnega protokola na LEGO MindStorm RCX robotih, namenjenim prikazu delovanja odložljivih omrežij in PRoPHET usmerjevalnega protokola. PRoPHET je usmerjevalni protokol, ki za usmerjanje zavojev med vozlišči v odložljivem omrežju uporablja zgodovino srečanj in prehodnosti. Dober način za emulacijo premičnih vozlišč je uporaba LEGO *MindStorms* RCX robotov, ki so programirani tako, da zadovoljijo naše komunikacijske potrebe in obidejo lokacijske ovire.

Diplomsko delo vsebuje pregled razlogov za nastanek odložljivih omrežij in opis težav, s katerimi se srečujemo pri usmerjanju podatkov ter implementacijo rešitev. Implementacija obsega majhno premično vozlišče, ki lahko z uporabo razvitih testnih orodij predstavi mehaniko PRoPHET protokola. Poleg premičnega vozlišča je bila razvita tudi predstavitvena platforma, ki se lahko uporablja za prikaz delovanja premičnega vozlišča v realnem času in/ali z analizo dnevnikov.

Rezultati dela kažejo, da je z uporabo LEGO MindStorms RCX mogoče na zelo jasen način pokazati delovanje odložljivih omrežij. Pokazalo se je tudi kar nekaj težav pri uporabi PRoPHET usmerjevalnega protokola za usmerjanje podatkov in glavne ugotovitve so zbrane v zaključku tega dela. Del ugotovitev, navedenih v diplomski nalogi, je postal osnova za nadaljnji razvoj PRoPHET usmerjevalnega protokola v sklopu projekta Sami Network Connectivity in prvo implementacijo odložljivega omrežja, namenjenega komunikaciji prebivalcev na skrajnem severu Švedske.

Ključne besede: odložljiva omrežja, PRoPHET, usmerjevalni protokol, LEGO, internet, prekrivna omrežja.

Abstract

Due to the ubiquitous network connectivity in the modern developed world it is easy to miss the need for networks that would enable communications in environments where network infrastructure is not present. Delay and disruption tolerant networks (DTN) enable communication in such environments as they do not need direct connectivity between nodes that intend to communicate.

This diploma presents an overview of an implementation that is intended to show how delay and disruption tolerant networks work; more specifically – it shows how PRoPHET routing protocol could be used on Lego MindStorms robots to demonstrate DTN networks in practice. PRoPHET routing protocol uses the history of encounters amongst network nodes to decide where it should forward the bundles of data. A good way to demonstrate the protocol mechanics is by using Lego MindStorms RCX robots, which are programmed and built in such a way to fulfill the communication and mobility needs.

The diploma explores the reasons for existence of the DTN networks and it analyzes problems that are present in the field of DTN routing algorithms. The final implementation consists of a small mobile node and a demonstration platform that can be used to demonstrate the PRoPHET protocol mechanics in real time or analyze it with the use of analysis and logging tools.

The results show that it is possible to demonstrate how a DTN network works using the LEGO MindStorms and at the same time uncover some of the issues faced when a DTN network implementation chooses to use the PRoPHET routing protocol. Some of the final conclusions and findings were the basis for further PRoPHET development as a part of the Sami Network Connectivity project. This development resulted in the first DTN network running PRoPHET protocol intended for use by the inhabitants of northern Sweden.

Key words: Delay Tolerant Networking, Disruption Tolerant Networking, PRoPHET, routing protocol, LEGO, computer networks, overlay networks.

1. UVOD

1.1. Nastanek projekta

Leta 2005 sem pričel z Erasmus izmenjavo v Švedskem mestu Lulea, kjer sem v sklopu predmeta računalniška omrežja pri prof. Kaustubhu S. Phanseju prvič spoznal termin odložljiva omrežja in protokol PRoPHET [6,7], ki je nastajal med doktorsko disertacijo prof. Andersa Lindgrena. Povabili so me k sodelovanju pri implementaciji protokola, ki za dostavo sporočil izkorišča premikanje vozlišč. Namen priprave implementacije je bila predstavitev delovanja protokola na konferenci Mobicom 2005 v Kölnu, Nemčija. Protokol po imenu PRoPHET (*Probabilistic Routing Protocol using History of Encounters and Transitivity*) je takrat obstajal zgolj v obliki osnutka RFC (kasneje je bil skupaj z mojimi popravki oddan na IETF [3] decembra 2006) in brez prave računalniške implementacije, ki bi lahko potrdila ali demonstrirala delovanje.

Rezultat mojega dela je bila prva uspešna implementacija odložljivega omrežja PRoPHET na LEGO MindStorms RCX[4], ki je bila osnova za nadaljnje raziskave s simulatorjem delovanja omrežja ter osnutek prve implementacije v resničnem svetu. V okviru projekta Sami Network Conectivity – SNC [10] je implementacijo v resničnem svetu dokončal Samo Grašič, študent Fakultete za računalništvo in informatiko, kar je tudi opisal v svojem diplomskem delu [2].

1.2. Razlogi za nastanek odložljivih omrežij

Morda se komu zdi, da pošiljanje podatkov preko računalniških mrež, ki doživljajo znatne zamude ali prekinitve, ne zahteva pomembne nove mrežne tehnologije, saj bi lahko zadostovali standardni internetni protokoli. Toda, obstajajo določene mrežne situacije, pri katerih internetni protokoli ne delujejo dovolj dobro, da bi se na njih lahko zanesli; zaradi tega je tudi potreben razvoj obravnavanega mrežnega pristopa.

Veliko okolij se srečuje z različnimi tipi prekinitev in zakasnitvami, ki jih po navadi povzročajo naprave na baterijsko napajanje, ki se izključijo zaradi varčevanja z elektriko. Težave nastanejo tudi pri premikanju tovrstnih naprav, kar povzroči, da so naprave med seboj preveč oddaljene za nemoten prenos informacij. V obeh primerih protokoli, ki implementirajo pristop shrani-in-posreduj, podobno kot deluje elektronska pošta, omogočajo

pomembne prednosti. Koncept mreženja, ki implementira pristop shrani-in-posreduj, se imenuje odložljivo mreženje (ang. *Delay Tolerant Networking - DTN*) in je glavna tema te diplome.

Odložljiva omrežja omogočajo izmenjevanje podatkov med vozlišči, ki so v danem trenutku lahko nepovezana ali pa sploh nikoli niso povezana – toda med njimi obstaja prehodna pot (preko vmesnih vozlišč) v realnem času. Primer tega je zunajzemeljska mreža satelitov za medplanetarno prenašanje sporočil. Če bi želeli poslati sporočilo med Zemljo in Marsom, bi lahko za ta namen uporabili satelite, ki krožijo okoli obeh planetov in druga telesa, ki bi jih postavili v orbite okoli sonca (na primer posredovalna postaja, ki bi krožila okoli sonca na ekvidistančni točki med planetoma). Zaradi kroženja satelitov (in zastiranja s strani nebesnih teles) v določenem trenutku mogoče ne bo možno predati sporočila, ko pa se sateliti ponovno postavijo, bo prenos sporočila možen.

Če primer razširimo na širše področje vgradnih sistemov, senzorskih omrežij in podobnih manj zmogljivih naprav – v smislu računske moči, zaloge energije ali hitrosti komunikacijskih vmesnikov – lahko hitro najdemo kar nekaj primerov, kjer je uporaba odložljivih omrežij smotrna.

Za določene izmed omenjenih sistemov je popolna izpraznitev baterije enaka uničenju. Tudi če bi lahko posamezno vozlišče popravili, bi bili lahko stroški popolne vzpostavitve vozlišča previsoki ali pa bi začasno nedelovanje vozlišča povzročilo veliko škodo. Zato je ohranjanje energije ena glavnih prioritet sistemov, ki so baterijsko napajani. Za primer lahko vzamemo brezžični sprejemnik in oddajnik, ki je eden največjih porabnikov energije na večini brezžičnih naprav. Za sprejemanje in posredovanje sporočil je namreč potrebno veliko energije in večja kot je razdalja med napravami, več energije potrebujemo. Po drugi strani pa je verjetno bolj cenovno učinkovita mreža z napravami, ki lahko med seboj komunicirajo na večjih razdaljah.

Enostavna rešitev za ohranjanje energije bi bil izklop radijskega oddajnika in sprejemnika, kar pa bi pomenilo, da je tudi naprava izključena in ne sprejema sporočil – tako pa lahko pride do zamud in (namernih) prekinitev v komunikaciji, saj bo vozlišče moralo 'počakati' s predajo sporočila, dokler se naslednje vozlišče v nizu ponovno ne vključi. Tovrstne zamude pa so lahko pomembne, saj je najbolj primeren način za nadziranje delovanja/nedelovanja radia povezan z neke vrste aplikacijskim ciklom vklapljanja in izklapljanja naprave. Tudi z morebitnimi dramatičnimi spremembami v baterijski tehnologiji bi bile težave z ohranjanjem

energije še vedno prisotne, saj po navadi želimo postaviti vozlišča za daljše obdobje, torej za več mesecev ali celo let. Iz tega razloga bodo sistemi morali vključevati še lastno pridobivanje energije preko, npr. sončnih celic in podobno.

Včasih se bo vozlišče tudi fizično premikalo in morda prišlo tudi na področje, kjer oddajanje ali sprejemanje ni mogoče; obstoječi omrežni protokoli lahko včasih, če prekinitev ni predolga, uspešno prestanejo tovrstne prekinitve, vendar jim po navadi spodleti. Končni uporabnik lahko to zazna (npr. v internetnem brskalniku) ali pa tudi ne (poštni odjemalec). Poleg težav z 'vidljivostjo' dostopnih točk se lahko zgodi, da se vozlišče giblje hitreje kot to omogoča infrastruktura. Trenutni infrastrukturni sistemi so zasnovani tako, da se obvezne posodobitve obravnavajo tako hitro kot je potrebno, vendar, če so dostopne točke z internetom povezane le posredno, lahko to povzroči zamude ali prekinitve.

Dostopne točke imajo lahko samo posredno dostopnost iz več razlogov, npr., zaradi cenejšega dostopanja do interneta ali pa zaradi spreminjanja lokacije (dostopna točka se lahko nahaja v različnih vozilih). Če to velja za določena infrastrukturna omrežja z dostopnimi točkami, je seveda situacija še slabša z omrežji, ki sploh nimajo infrastrukture. V tovrstnih *adhoc* omrežjih je lahko verjetnost, da bo povezava med dvema točkama uspešno vzpostavljena, zelo majhna. Z upoštevanjem vsega navedenega lahko vidimo eno od glavnih razlik med odložljivimi (DTN) omrežji in drugimi pristopi.

Zadnja ključna ugotovitev, ki velja za vgradne sisteme, je, da tovrstni sistemi veliko časa pravzaprav nič ne počnejo. Imeti povezano napravo samo zaradi tega, da lahko do nje dostopamo, je energijsko zelo potratno. Tako se pojavi potreba po sistemu, ki se sam ugasne in se 'zbudi' šele takrat, ko se nekaj dogaja. Najbolj pogosta rešitev za ta problem je vnaprej določen alarmni klic (ta ima napajanje običajno ločeno od sistema), ki sistem zbudi. Veliko procesorjev ima več možnosti delovanja sistema, podobno kot pri osebnih računalnikih. To so: delovanje, stanje pripravljenosti, hibernacija in izključitev (vendar ne gre za popolno izključitev, saj veliko naprav še vedno deluje, če so priključene). Neodvisno od načina ugašanja in prižiganja sistema bo omrežje zagotovo doživelo zamude in prekinitve na skoraj enak način, kot če bi prižigali in izključevali radijski oddajnik in sprejemnik.

1.3. Obstoječe rešitve

Na svetu že danes obstaja kar nekaj rešitev, ki uspešno implementirajo koncepte odložljivih omrežij.

- **Države v razvoju**

  V manj razvitih državah se pogosto zgodi, da enostavno ni možnosti za vzpostavitev moderne komunikacijske infrastrukture, čeprav je ta izjemno pomembna za dviganje kakovosti življenja prebivalcev. Primer uporabe najdemo v projektu Wizzy digital courier [11], kjer za dostavo informacij v ruralne šole Južne Afrike uporabljajo USB ključe in motorna kolesa. Spletni zahtevki in e-pošta se v šoli naložijo na USB ključ, ki ga odpeljejo do mesta, kjer je mrežni priklop možen. Tam se podatki sinhronizirajo in odpeljejo jih nazaj v šolo.

- **Senzorska omrežja**

  Verjetno najbolj znan primer senzorskega omrežja, ki za komunikacijo uporablja odložljiv protokol, je projekt ZebraNet [13], ki ga vodijo na ameriški univerzi Princeton. Namen projekta so raziskave gibanja populacije zeber v afriški divjini. V ta namen so razvili ovratnico, ki se vsakih nekaj minut prižge in zabeleži GPS lokacijo. Ovratnica vsakih nekaj ur prižge svoj sprejemnik in oddajnik z namenom izmenjave zbranih informacij z drugim zebrami, ki so prav tako opremljene z ovratnico. Ideja je bila, da vse te zbrane informacije potem poberejo z občasnimi prehodi afriške divjine, brez iskanja vsake izmed zeber, ki so jo opremili z oddajnikom.

Obstajajo tudi protokoli, nekateri zgodovinski, drugi aktualni, ki so potencialno pomembni pri razmišljanju kako doseči zahteve odložljivih protokolov. Verjetno največkrat omenjen je IP protokol "poštni golob" ali "golob pismonoša" [12]. Implementacija tega protokola je dejansko pokazala, da lahko internetni protokol (IP) – ne pa tudi TCP – uspešno deluje v mnogih nepričakovanih okoljih.

1.4. Zavojni protokol

Odločilno dejstvo za nastanek odložljivih omrežij je, da je *Transmission Control Protocol* (TCP) [9] tisti internetni protokol, ki ne deluje v razmerah, kjer nimamo zanesljive in direktne povezave (lahko tudi prek drugih vozlišč).

Da bi TCP protokol deloval, mora med vozliščema obstajati vsaj ena stalna dvosmerna povezava (povezava mora biti dvosmerna, saj TCP zahteva potrjevanje prenesenih podatkov), ki omogoča izmenjavo podatkov z relativno majhnimi zamiki (tipične TCP implementacije prekinejo povezavo, če dve minuti ni izmenjave podatkov) in z relativno visoko zanesljivostjo (na nezanesljivih povezavah prihaja do veliko zahtev po ponovnih prenosih podatkov). Tako TCP ni primeren za komunikacijo po omrežjih brez stalne povezave med vozlišči, saj je za uspešno komunikacijo potrebno prenos podatkov pogosto potrjevati.

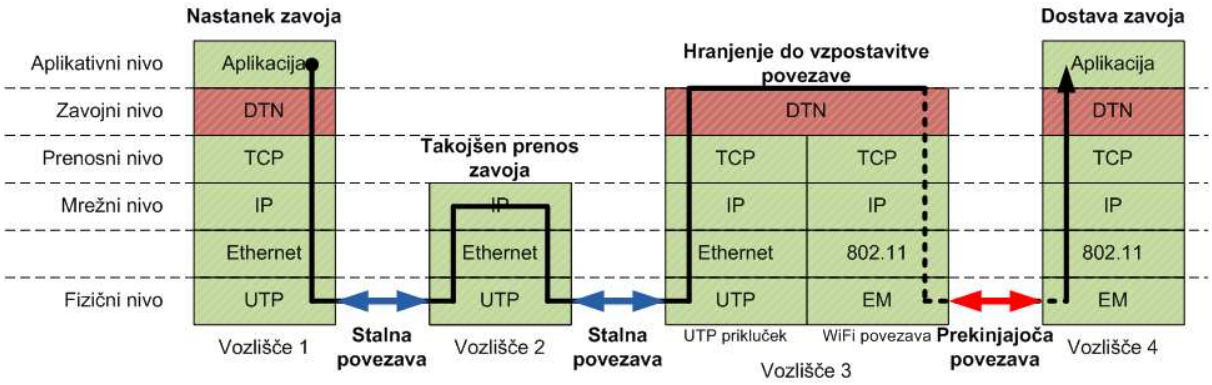
TCP je osnova za vse glavne internetne aplikacije, ki jih najpogosteje uporabljamo: elektronsko pošto, splet in mnoge druge. Torej je potreba po novih protokolih, ki delujejo v situacijah, v katerih TCP odpove, resnična.

Izkazalo se je, da obstaja veliko aplikacij, ki imajo korist od uporabe odložljivega mreženja. Vendar ne gre samo za aplikacijske zahteve, prav tako potrebujemo tudi razumne načine kako bomo zadostili tem zahtevam. Zaradi opisanih potreb se danes razvijajo nekateri protokoli, ki bi lahko čez nekaj let predstavljali standardne protokole kot je sedaj, npr. *TCP*.

Eden izmed teh se imenuje zavojni protokol (ang. *bundle protocol*), ki v osnovi deluje skorajda enako kot današnja elektronska pošta, vendar je njegov cilj integracija v druge aplikacije in ne delovanje kot samostojna aplikacija. Pristop je v osnovi takšen, da vzame enoto aplikacijskih podatkov in jo zapakira skupaj z vsemi potrebnimi nadzornimi podatki v zavoj, ki je zelo podoben elektronskemu sporočilu – elektronski pošti. Zavoj nato posreduje po določeni poti, ki jo sestavlja skupek vmesnih strojev, od katerih lahko vsak shrani sporočilo za določen čas.

Zavojni protokol je implementiran kot dodaten "zavojni nivo" (ang. *bundle layer*), ki je postavljen med aplikacijskega in ostale mrežne nivoje. Zavojni protokol izmenjuje podatke v obliki zavojev (ang. *bundle*), ki so sestavljeni iz glave zavoja (naslov pošiljatelja, naslov prejemnika, …) in podatkov, ki jih je posredovala izvorna aplikacija.

Zavojni protokol je primer tega, kar na splošno imenujemo prekrivno omrežje (ang. *overlay network*). Lahko ga izvajamo na trenutnih internetnih protokolih, pa tudi preko bolj ekskluzivnih protokolov, ki jih uporabljajo vesoljska plovila ali takšnih, ki jih predlagajo raziskovalci za obravnavanje kompleksnih senzorskih omrežij ali v drugih zahtevnih okoljih.



Slika 1: Umestitev zavojnega nivoja v OSI model

Na sliki 1 je prikazana umestitev zavojnega nivoja v OSI (ang. *Open Systems Interconnect*) model modularne zgradbe komunikacijskih protokolov. Podatke, ki jih želi aplikacija na vozlišču 1 dostaviti vozlišču 4, shrani v zavoj z naslovom prejemnika. Zavojni nivo zavoj posreduje prek nižjih nivojev do povezanega vozlišča 3, kjer počaka do ponovne vzpostavitve povezave med vozlišči 3 in 4. Zavoj se prenese v zavojni nivo vozlišča 4, kjer aplikacija prebere podatke, ki so shranjeni v poslanem zavoju.

1.5. Ključne lastnosti

Zavojni protokol vključuje dva ključna pristopa, ki omogočata delovanje v okoljih, kjer klasični protokoli odpovedo.

Prva ključna lastnost je pristop shrani-in-posreduj. Klasični omrežni protokoli za uspešno komunikacijo med dvema vozliščema nujno potrebujejo znano pot (lahko tudi prek drugih vozlišč), saj vsako izmed vmesnih vozlišč prihajajoči paket na podlagi pravil posreduje naprej po ustrezni poti. Če ustrezna pot ni znana, se paket zavrže kot nedostavljiv. Odložljiva omrežja uporabljajo pristop shrani-in-posreduj, ki nedostavljive podatke shrani z namenom naknadnega posredovanja naslovniku, ki se v omrežje poveže kasneje.

Druga ključna lastnost zavojnega protokola je nepogovornost le-tega. TCP je primer pogovornega protokola, saj tudi manjšo količino podatkov razdeli v veliko število majhnih paketov in za vsakega izmed njih pričakuje potrditev prenosa. Takšno delovanje zahteva dvosmerno povezavo, kar je v okoljih, za katera so namenjeni zavojni protokoli, neizvedljivo. Zavojni protokol pa, nasprotno, zbere vse podatke v en velik zavoj in le-tega, brez zahteve po vzpostavitvi povezave, pošlje prek prekrivnega omrežja. Nižji nivoji (npr. TCP) poskrbijo za to, da je celoten zavoj prenesen do naslednjega vozlišča, na katerem teče zavojni protokol, le-to pa prejema zavoja ne potrjuje. Če podatkov ne more posredovati naslovniku, jih shrani, ko

pa ima možnost za posredovanje podatkov naslednjemu zavojnemu vozlišču, le-te posreduje. Ko je zavoj uspešno dostavljen, prejemnik pošlje potrdilo o prejemu zavoja, s katerim obvesti ostala vozlišča, da lahko prenehajo s hranjenjem zavoja in pošiljatelja, da je bil zavoj uspešno dostavljen. Sam protokol za uspešno delovanje ne zahteva pošiljanja potrdila prejema, le-to je pomembno zgolj zaradi odstranjevanja uspešno dostavljenega zavoja iz vozlišč v odložljivem omrežju.

1.6. Težave v odložljivih omrežjih

Zavojni protokol za svoje delovanje (dostavo podatkov) ne zahteva nobenih povratnih informacij s strani posrednikov ali prejemnika – dokler prejemnik prejema zavoja ne potrdi, le-ta ostaja shranjen na vozliščih. Ta lastnost protokola je problematična s stališča razpoložljivega prostora za hranjenje zavojev, zato zavojni protokol implementira razne strategije, ki skrbijo za pametno odstranjevanje zavojev iz shramb v primeru pomanjkanja prostora – ob hkratnem zagotavljanju dostave zavoja.

Poleg težav s strategijo odstranjevanja zavojev se pojavi ključno vprašanje kako pametno usmerjati podatke v omrežjih, kjer vnaprej ni znano katera pot je primerna za dostavo – saj je možno, da v trenutku pošiljanja zavoja le-ta še ne obstaja. To vprašanje pridobi še dodatno pomembnost zaradi dejstva, da slaba odločitev o posredovanju zavoja direktno vpliva na število dvojnikov zavoja – slabo usmerjen paket bo zasedal prepotreben prostor na drugem vozlišču. Za uspešno delovanje odložljivega omrežja je ključen premislek o primernem usmerjevalnem protokolu.

2. USMERJANJE PODATKOV V ODLOŽLJIVIH OMREŽJIH

2.1. Težave z usmerjanjem podatkov

Obstaja veliko protokolov za usmerjanje, ki se ukvarjajo s težavo usmerjanja podatkov v odložljivih omrežjih. Večina jih deluje, če jih oskrbujemo z neomejenimi resursi – prepustnost povezav, količina pomnilnika in zaloga energije. Pri resničnih posredno povezanih mrežah pa so resursi omejeni (ali celo nezadostni). V praksi se v takih omrežjih težava usmerjanja podatkov rešuje z uporabo statičnih usmerjevalnih tabel, ki jih skrbniki omrežja urejajo ročno. Takšen pristop je za široko uporabo neprimeren, saj s povečevanjem velikosti omrežja problem hitro postane neobvladljiv. S tem namenom nastaja kar nekaj pristopov k usmerjanju podatkov v odložljivih omrežjih, ki zmanjšajo porabo resursov s pametnim razmislekom komu posredovati določen paket.

Primer protokola, ki odlično deluje v primeru neomejenih resursov, je epidemično usmerjanje, ki deluje po principu epidemije – hitrega širjenja prenosljive bolezni. Epidemija se začne z okužbo prvega nosilca in ta ob vsakem srečanju z drugo osebo le-to okuži (z boleznijo). Vsaka okužena oseba deluje kot posrednik bolezni in – tako kot prvi okuženec – okuži ostale, še ne obolele osebe. Čez določen čas so vse osebe, ki prihajajo v kontakt z drugimi osebami, okužene. Natanko isti pristop prevzame epidemično usmerjanje, ki novonastalo sporočilo na viru sporočila posreduje vsem ostalim vozliščem, ki se z virom sporočila srečajo. Vsako izmed vozlišč postane nosilec sporočila in le-tega posredujejo naprej vsem vozliščem, ki jih srečajo. Čez čas se bo sporočilo razširilo tudi do končnega prejemnika sporočila.

V primeru neomejenih resursov je epidemično pošiljanje najbolj učinkovit protokol za dostavo sporočil s stališča dostave sporočil, saj zagotavlja, da bo sporočilo dostavljeno prejemniku, če do prejemnika obstaja pot prek vmesnih vozlišč. Če pa epidemično usmerjanje uporabimo v omrežju manjših brezžičnih naprav, se protokol izkaže za manj primernega, še posebej ko velikost omrežja in količina novih sporočil, ki nastane na vsakem izmed vozlišč, naraščata.

Težave bom pokazal z uporabo epidemičnega usmerjanja v sklopu senzorskega omrežja – omrežja majhnih vgradnih naprav, namenjenih zbiranju podatkov. Vsako vozlišče posreduje sporočilo vsem ostalim vozliščem in hitro se zgodi, da vsaka naprava hrani ogromno sporočil, saj shrani vsako prejeto sporočilo. V primeru senzorskih omrežij je to neizvedljivo, saj so omejena s količino prostora, namenjenega hrambi sporočil. Situacija se še poslabša, ker se vsako sporočilo pošilja določeno količino časa. Najprej je to pomembno s stališča avtonomije naprave – brezžična komunikacija je izjemno energetsko potratna dejavnost in ker se izmenjujejo vsa sporočila, ki jih prejemnik še nima, se med izmenjavo porabi veliko energije. Drugi faktor, ki zmanjšuje učinkovitost, je dejstvo, da ni nujno, da so povezave med vozlišči stalne (vozlišča se lahko premikajo), kar pomeni, da ima vozlišče za izmenjavo podatkov omejen časovni okvir. Če je sporočil veliko, obstaja verjetnost, da se ne bodo vsa uspešno izmenjala.

Zgornje težave rešimo z vpeljavo neke metrike, ki določa katera sporočila je vredno posredovati določenemu prejemniku. Tako se zmanjša količina posredovanih sporočil, kar zmanjša potrebe po kakovosti oz. hitrosti povezav in zmanjša potrebe po pomnilniku in energiji, ki jo naprava porabi za izmenjevanje sporočil.

2.2. PRoPHET usmerjevalni protokol

PRoPHET usmerjevalni protokol vpelje metriko verjetnosti dostave za odločanje o primernosti prejemnika za sprejem sporočila. Metrika verjetnosti izkorišča dejstvo, da se v resničnih okoljih prava vozlišča (uporabniki) ne premikajo naključno, ampak se srečanja med vozlišči ponavljajo.

Vzemimo za primer študente Fakultete za računalništvo in informatiko. Študenti istega letnika se pogosto srečujejo na skupnih predavanjih, študentje različnih letnikov pa zgolj v skupnih interesnih skupinah. To dejstvo izkoristimo za odločanje komu želimo posredovati sporočilo – če ima nek študent iz drugega letnika sporočilo za študenta prvega letnika, je smotrno, da sporočila ne posreduje vsem ostalim študentom svojega letnika, ampak samo tistim študentom, ki se družijo s študenti iz prvega letnika. Tako smo uporabili zgodovino srečanj za postavitev metrike, ki bo odločala kdo je primeren posrednik sporočila.

PRoPHET verjetnostne tabele, ki jih uporabljamo za izbor zavojev, ki jih bomo odposlali, so zasnovane na treh osnovnih enačbah – na *zgodovini srečanj*, *prehodnosti med vozlišči* in *staranju verjetnostnih tabel*. Vsakič, ko se dve vozlišči srečata, PRoPHET odredi določeno

dostavno verjetnost za ponovno srečanje (zgodovina srečanj). Prav tako se dodeli določena dostavna verjetnost, če lahko določeno vozlišče dosežemo prek drugega vozlišča (prehodnost oz. tranzitivnost) – to ugotovimo s preverjanjem vpisov v verjetnostnih tabelah vozlišč, ki se srečujejo. Poskrbeti je treba tudi za zmanjševanje dostavne verjetnosti zavoja do določenega vozlišča, če se dve vozlišči dalj časa ne srečata – v tem primeru bi verjetnost posredovanja zavoja med njima morala biti zelo nizka (stara).

2.3. PRoPHET verjetnostne tabele in izračun dostavnih verjetnosti

PRoPHET verjetnostne tabele vsebujejo podatke o identifikaciji vozlišča in dostavni verjetnosti do tega vozlišča za vsa poznana vozlišča. Ko se dve vozlišči srečata, poskrbita za izmenjavo svojih verjetnostih tabel in izračun novih vrednosti na podlagi prej omenjenih treh enačb.

- **Izračun dostavne verjetnosti na podlagi zgodovine srečanj**

$$\mathbf{P'(A,B) = P(A,B) + (1 - P(A,B)) * P_{srečanje}} \quad (1)$$

$$\mathbf{0 \leq P_{srečanje} \leq 1}$$

P'(A,B) je nova dostavna verjetnost med vozliščem A in drugim vozliščem B. Izračunamo jo na podlagi stare dostavne verjetnosti **P(A,B)** med vozliščema in konstanto $\mathbf{P_{srečanje}}$, ki določa vrednost srečanja dveh vozlišč. Bolj ko cenimo direktna srečanja med vozlišči, bližje vrednosti 1 jo določimo.

- **Izračun dostavne verjetnosti z upoštevanjem prehodnosti med vozlišči**

$$\mathbf{P'(A,C) = P(A,C) + (1 - P(A,C)) * P(A,B) * P(B,C) * \beta} \quad (2)$$

$$\mathbf{0 \leq \beta \leq 1}$$

P' je nova dostavna verjetnost med vozliščem A in drugim vozliščem C, s katerim trenutno nismo v stiku, je pa podatek zapisan v verjetnostnih tabelah vozlišča B, s katerim smo trenutno v stiku. Izračunamo jo na podlagi stare dostavne vrednosti **P(A,C)** za vozlišče C in stare dostavne vrednosti **P(B,C)**, ki smo jo prejeli od vozlišča B ter konstante **β**, ki vrednoti pomembnosti prehodnih (tranzitivnih) povezav.

- **Staranje verjetnostnih tabel**

$$\mathbf{P'(A,B) = P(A,B) * \gamma^{K}} \quad (3)$$

$$\mathbf{0 \leq \gamma \leq 1}$$

Pred izmenjavo verjetnostnih tabel izvedemo operacijo staranja le-teh, saj se lastnosti omrežja – vzorci srečevanj – skozi čas spreminjajo, oz. je bilo lahko srečanje zgolj naključno. Staranje je nujno potrebno, saj v nasprotnem primeru dostavne verjetnosti konvergirajo proti 1.

P'(A,B) je nova dostavna verjetnost med vozliščema A in B, ki jo izračunamo na podlagi stare dostavne verjetnosti **P(A,B)**, konstante staranja **γ** in številom pretečenih časovnih period **K**.

Izbor konstant v enačbah je ključnega pomena za učinkovito delovanje PRopHET usmerjevalnega protokola. V osnutku PRoPHET protokola so bile predlagane naslednje vrednosti za nastavitev konstant:

$$\mathbf{P_{srečanje} = 0.75;\ \beta = 0.25;\ \gamma = 0.98} \quad (4)$$

Med delom na projektu se je izkazalo, da v resnici pravila za določitev konstant ni, saj so le-te močno odvisne od lastnosti samega omrežja. Zaradi tega je nastalo kar nekaj težav, ki so opisane v poglavju 5.

2.4. Posredovanje zavojev

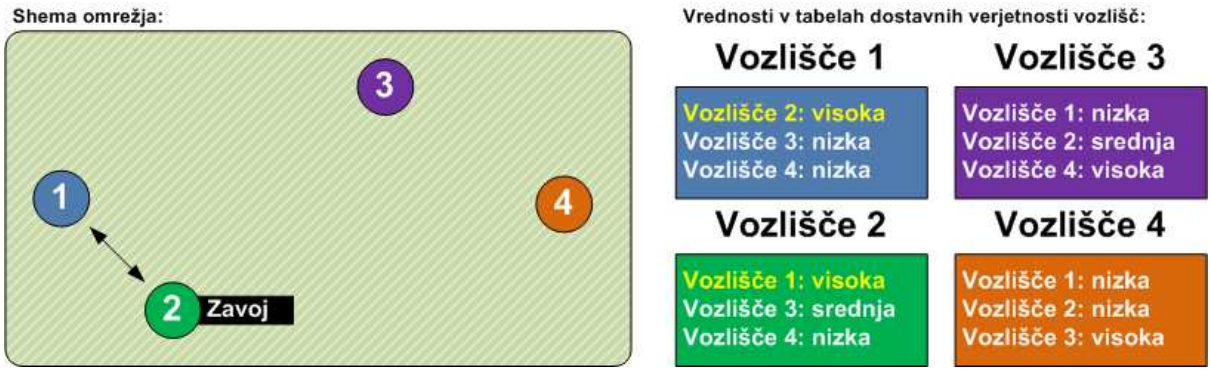
Z uporabo dostavnih verjetnosti lahko vozlišče odloča komu naj posreduje posamezen zavoj. Možnih je več strategij posredovanja, sam sem se odločil za t.i. strategijo GRTR, po kateri vozlišče posreduje zavoj samo če je dostavna verjetnost, shranjena v tabeli dostavnih verjetnosti vozlišča, s katerim izmenjujemo podatke, za ciljno vozlišče večja od tiste, ki je shranjena v naši tabeli dostavnih verjetnosti.

$$\mathbf{P(B,C) > P(A,C)} \quad (5)$$

GRTR pogoj za posredovanje zavojev – Vozlišče A posreduje zavoj (namenjen vozlišču C) vozlišču B samo v primeru, da je dostavna verjetnost vozlišča C v tabelah vozlišča B strogo višja od dostavne verjetnosti vozlišča C v tabelah vozlišča A.

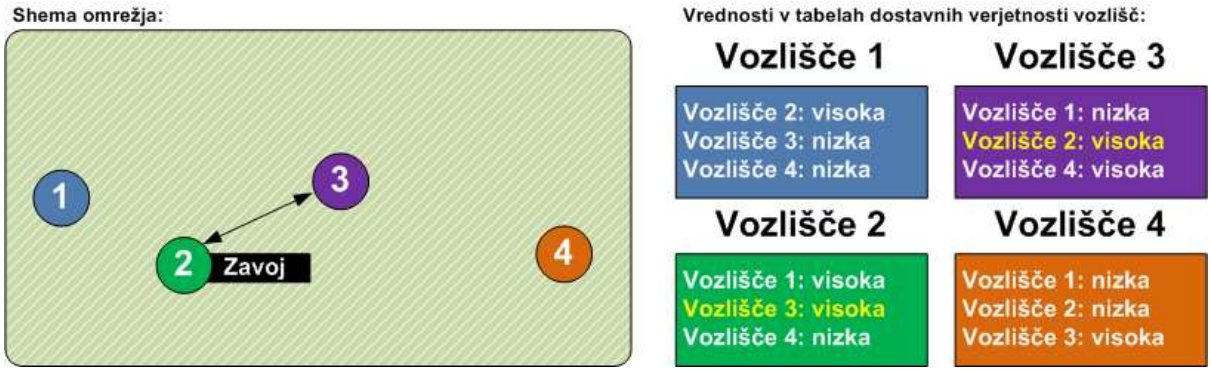
GRTR ni edina strategija za posredovanje zavojev, saj lahko upoštevamo še frekvenco srečanj, ki omejijo najnižjo dovoljeno verjetnost za posredovanje in so bolj natančno opisane v RFC dokumentu PRoPHET protokola [6].

2.5. Prikaz delovanja PRoPHET usmerjanja



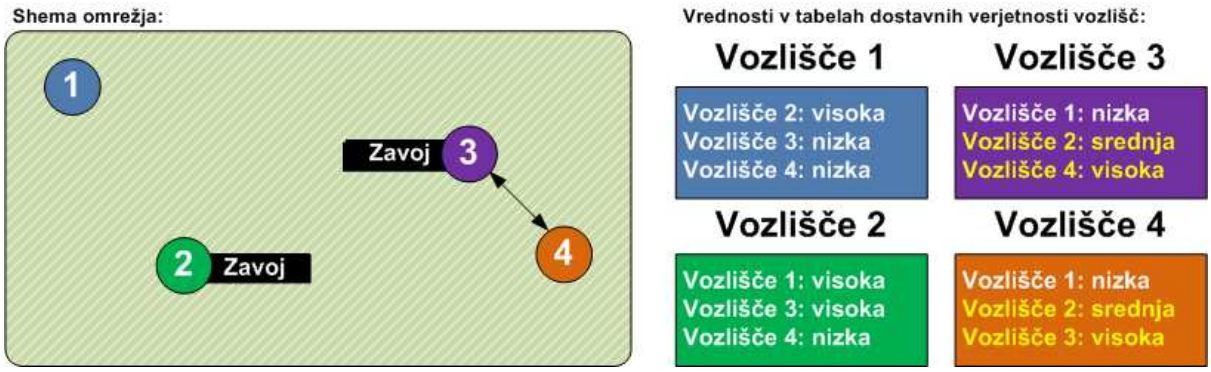
Slika 2: Prikaz delovanja PRoPHET usmerjanja – prvi korak

V omrežju se nahajajo 4 vozlišča. Aplikacija na vozlišču 2 je ustvarila zavoj za vozlišče 4. Vozlišči 1 in 2 se srečata ter izmenjata podatke v tabelah dostavnih verjetnosti. Vozlišče 2 ne posreduje zavoja vozlišču 1, saj je vrednost v njegovi dostavni tabeli nižja od vrednosti, ki je shranjena v tabeli vozlišča 2.



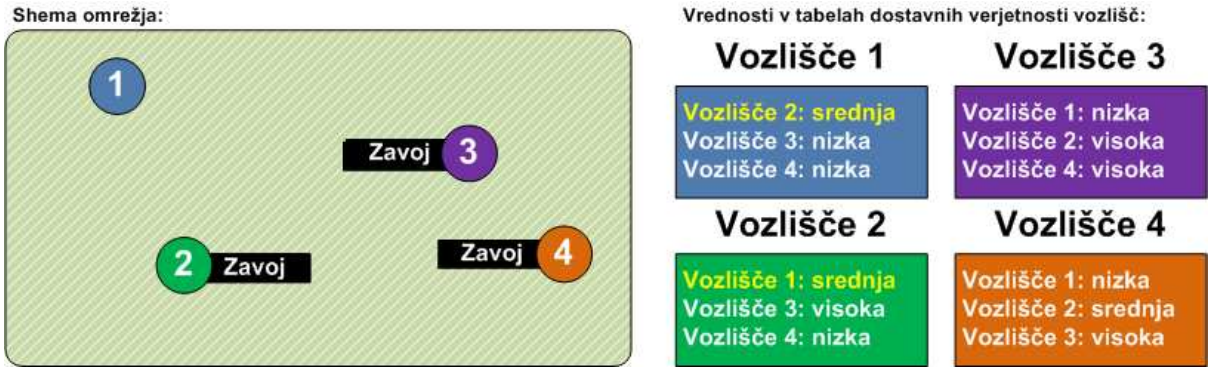
Slika 3: Prikaz delovanja PRoPHET usmerjanja – drugi korak

Zaradi premikanja vozlišč se vozlišči 2 in 3 dovolj približata, da se med njima vzpostavi povezava. Izmenjata vsebino tabel dostavnih verjetnosti. Vozlišče 2 se na podlagi prejete tabele odloči, da je vozlišče 3 primeren kandidat za sprejem zavoja in ga posreduje. Zaradi lastnosti zgodovine srečanj obe vozlišči prilagodita verjetnosti v lastnih tabelah dostavnih verjetnosti.



Slika 4: Prikaz delovanja PRoPHET usmerjanja – tretji korak

Vozlišči 3 in 4 uspešno vzpostavita povezavo in izmenjata tabeli dostavnih verjetnosti. Vozlišče 3 prilagodi vnos za vozlišče 4 (upoštevanje zgodovine srečanj) in vozlišče 2 (prehodnost med vozlišči). Vozlišče 4 prilagodi vnos za vozlišči 3 in 2. Vozlišče 3 posreduje zavoj vozlišču 4, saj je naslovnik zavoja.



Slika 5: Prikaz delovanja PRoPHET usmerjanja – četrti korak

Od zadnjega srečanja vozlišča 1 z vozliščem 2 je preteklo kar nekaj časa. Sproži se postopek staranja vnosov, v katerem vozlišči 1 in 2 zmanjšata dostavne verjetnosti.

3. IZBIRA TEHNOLOGIJE ZA PRIKAZ DELOVANJA PROTOKOLA PROPHET

3.1. Cilji implementacije

Sam izbor tehnologij je kritično določen z zastavljenimi cilji in pred začetkom razvoja so bili zastavljeni tisti, ki jih mora prva implementacija protokola PRoPHET izpolniti, da bo uspešna.

Primaren cilj je bil priprava implementacije, ki omogoča dobro demonstracijo delovanja protokola, saj je bil eden izmed namenov projekta demonstracija delovanja udeležencem konference Mobicom 2005. Ker je sam projekt nastajal v sklopu projekta Sami Network Conectivity, sem želel pokazati kako bi s pomočjo PRoPHET protokola več vasi na področju severne Švedske komuniciralo. Vasi so razpršene po širokih prostranstvih, kjer komunikacijske infrastrukture ni na voljo, se pa ljudje vsak dan med vasmi gibljejo s pomočjo motornih sani.

Predstavitev in pojasnitev delovanja protokolov je kompleksna naloga, zato je bilo potrebno pripraviti dobro demonstracijsko okolje, ki bi tudi neizkušenim osebam omogočilo grafično predstavitev delovanja protokola. Takšno demonstracijsko okolje mora vsebovati premična vozlišča, ki ponazarjajo gibanje vozlišč (na primer voznikov motornih sani) in seveda nek grafični uporabniški vmesnik, ki omogoča spremljanje aktivnosti in delovanja vozlišč v realnem času. Uporabnik ima možnost interakcije z mobilnimi vozlišči (pošiljanja sporočil skozi mrežo mobilnih vozlišč). Predvideni so bili tudi posebni paketi, ki prikazujejo paketne poti skozi mrežo, pripomočki za priklic dnevnikov in analizo ter nek pripomoček za vizualizacijo delovanja omrežja.

Poleg same demonstracije je bil zastavljen tudi cilj, da sama implementacija protokola deluje tako kot je določeno v specifikaciji, saj sem želel preveriti veljavnost argumentov v originalnem osnutku RFC protokola PRoPHET [6]. Ker je bilo jasno, da bo moje delo osnova za nadaljnje implementacije, sem si zastavil še cilj, da bo sama implementacija prenosljiva med raznimi sistemi, ki bi jih lahko uporabili za komunikacijo.

3.2. Demonstracija delovanja in Lego MindStorms

Ker PRoPHET protokol izkorišča mobilnost vozlišč za dostavo informacij, je bilo potrebno uporabiti majhno platformo, ki bo zmožna avtonomnega premikanja, bo široko razširjena, dobro dokumentirana in bo omogočala razvoj v enem izmed razširjenih programskih jezikov.

Izmed vseh možnih rešitev, ki so bile na voljo v času nastanka projekta, se idealu še najbolj približa rešitev LEGO MindStorms RCX.

LEGO MindStorms RCX je eden izmed izdelkov danskega proizvajalca LEGO, ki je namenjen izdelavi robotov s pomočjo LEGO kock. Glavna komponenta sistema je RCX modul, v katerega je vgrajen programabilen mikrokrmilnik. V začetku se je RCX modul lahko programiral zgolj s pomočjo preprostega didaktičnega orodja, ki je bil priložen kompletu, vendar so kasneje vztrajni posamezniki na samo platformo prenesli množico klasičnih programskih jezikov: Java – leJOS [5], C/C++ – brickOS [1], Visual Basic, Fortran – pbForth [8]. Prenos klasičnih programskih jezikov je omogočil dosti bolj širok spekter rešitev, za katere je RCX modul uporaben.

Poleg programabilnega mikrokontrolerja so v samo platformo vgrajeni tudi vmesniki za priklop periferije – v sklopu Lego programa obstaja širok nabor senzorjev in motorjev, ki jih lahko priklopimo na RCX. Ključnega pomena je tudi vgrajeni IR komunikacijski vmesnik, ki je uporaben za medsebojno komunikacijo med RCX moduli. Poleg možnosti komunikacije med samimi RCX moduli je obstajala tudi možnost komunikacije z računalnikom s pomočjo priloženih komunikacijskih stolpov – Lego gradnikov, ki so imeli vgrajen IR senzor in oddajnik ter možnost priklopa na osebni računalnik prek zaporednega ali USB vmesnika.

V uporabo sem dobil šest paketov Lego MindStorms dveh različnih revizij – razlikovale so se po količini elementov ter po priloženi strojni opremi. Lego MindStorms RCX 1.0 so vsebovali komunikacijski stolp s priklopom na zaporedni vmesnik, Lego MindStorms RCX 2.0 pa stolp z USB vmesnikom. Do razlik je prihajalo tudi pri sami strojni opremi, vgrajeni v module, kar je pomenilo, da se vsaka izmed revizij kompleta pri komunikaciji obnaša malo drugače – novejša revizija je občutno bolj občutljiva na motnje.

3.3. Izbor programskega okolja

Pri razvoju implementacije protokola je bil cilj doseči enostavno prenosljivost med različnimi platformami, na katerih se bo le-ta uporabljala. Pri izboru programskega jezika sem bil omejen z naborom programskih jezikov, ki so na voljo za uporabo na Lego RCX in s ciljem, da uporabim enako implementacijo na vseh platformah, ki jih bom uporabljal. V osnovi je to pomenilo programski jezik, ki bo omogočal izvajanje enake programske kode na osebnem računalniku in Lego RCX modulu.

Že prej je bilo omenjeno, da obstaja množica alternativ pri izboru programskega jezika, ki bi bile lahko uporabne pri pripravi implementacije na RCX modulu. Po pregledu možnosti sem se odločil za uporabo projekta leJOS, v sklopu katerega je na samem RCX modulu implementiran majhen Java virtualni stroj, ki omogoča izvajanje klasičnih Java programov z določenimi omejitvami.

Uporaba leJOS omogoča, da modul RCX programiramo s pomočjo objektno orientiranega jezika, ki nudi podporo izvajanju niti, podpira kompleksne podatkovne strukture, vključuje knjižnico za matematične operacije in odlično dokumentirano knjižnico za delo s perifernimi enotami, ki jih lahko priklopimo na RCX modul.

Knjižnica Robotics API omogoča:

- nadzor nad gumbi, ki so na samem RCX modulu;
- izpis podatkov na vgrajenem znakovnem prikazovalniku;
- nadzor motorjev, ki robotu omogočajo gibanje;
- branje stanja senzorjev – senzor pritiska, temperature, svetlobe in kota;
- komunikacijo s pomočjo vgrajenega IR vmesnika.

Dodaten plus je bilo dejstvo, da je Java izjemno popularen programski jezik, kar je s stališča cilja priprave referenčne implementacije izjemno pomembno, saj naj bi bila pripravljena implementacija tako lažja za uporabo v ostalih projektih.

Kot bo prikazano v poglavju 5, je odločitev za uporabo Jave povzročila kar nekaj preglavic, ki v fazi izbora niso bile predvidene.

3.4. Grafično orodje za demonstracijo delovanja

Predstavitev in pojasnitev delovanja protokolov je kompleksna naloga, zato je bilo potrebno dobro demonstracijsko orodje, ki bi tudi neizkušenim osebam omogočilo grafično predstavitev delovanja protokola. Cilj je bil pripraviti grafični uporabniški vmesnik, ki omogoča spremljanje delovanja protokola in delovanja vozlišč v realnem času.

Funkcionalne zahteve:

- Uporabnik/demonstrator naj ima možnost interakcije z mobilnimi vozlišči – omogočiti je potrebno pošiljanje sporočil skozi mrežo mobilnih vozlišč.
- Uporabnik/demonstrator naj ima možnost pošiljanja posebnih paketov, ki prikazujejo paketne poti skozi mrežo.
- Uporabnik/demonstrator naj ima možnost zajema dnevnikov delovanja, shranjenih na posameznem Lego RCX modulu. Na voljo naj bodo orodja za kasnejšo analizo. Opcijsko naj bo na voljo tudi možnost vizualizacije potovanja paketa.

Pri izbiri okolja za pripravo grafičnega orodja za demonstracijo delovanja sem bil omejen z izbranim programskim jezikom in orodji, ki so mi v sklopu le-tega bila na voljo. Po pregledu možnosti sem si izbral uporabo Java Swing-a. Java Swing je nabor programskih vmesnikov *(API)*, ki je namenjen izgradnji programov z grafičnim vmesnikom v Java programskem jeziku.

4. IMPLEMENTACIJA

4.1. RCX vozlišče

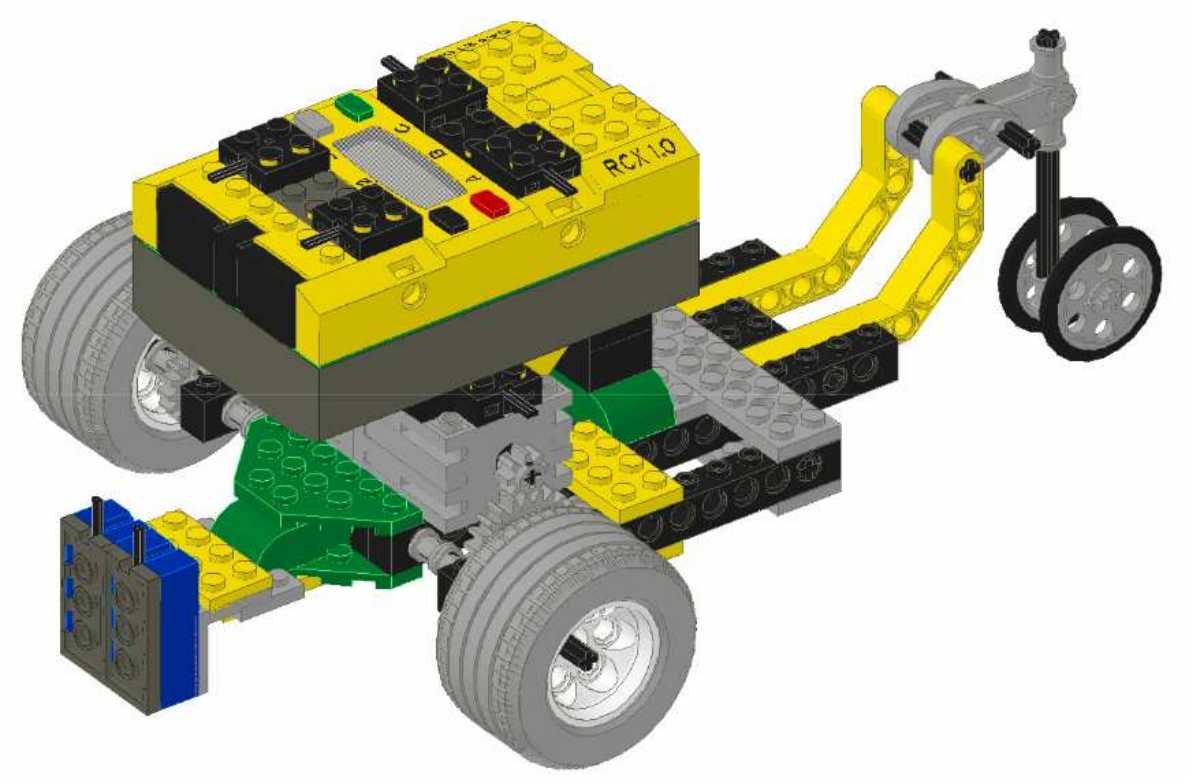
Pri razvoju RCX vozlišča so bile preizkušene različne rešitve, ki bi sledile ideji, da naj se vozlišča gibljejo po bolj ali manj 'konstantni' poti. Prvi preizkusi so pokazali, da bi IR komunikacija znala biti problematična, saj zahteva, da se senzorja dveh RCX vozlišč med komunikacijo vidita. Zato je bila potrebna rešitev, ki bi zagotovila predvidljivost gibanja in neke vrste nadzor nad tem kje se bodo roboti srečevali.

Na koncu je bila izbrana rešitev – dizajn "robota, ki sledi liniji," saj omogoča predvidevanje poti (narisani poti je robot natančno sledil) in omogoča tudi predvidevanje krajev, kjer se bo odvila komunikacija podatkov. Na krajih, kjer naj bi roboti komunicirali, so le-ti potovali po dolgi ravni poti, tako da so se lahko IR senzorji poravnali, kar je omogočilo uspešno komunikacijo. Lahko bi trdili, da je uporaba tovrstne rešitve zelo deterministična in nasprotuje naravni naključnosti, ki bi jo PRoPHET sicer moral obvladati – toda dejanski testi so pokazali, da temu ni tako. Kljub uporabi vnaprej določenih poti in ravnin za komunikacijo, je bila celotna mreža daleč od tega, da bi bila deterministična, predvsem zaradi težav z IR komunikacijo. IR senzor namreč ni zagotavljal, da bo v primeru direktne vidne poti tudi komunikacija potekala brez težav. Nasprotno, sam IR senzor je bil zelo občutljiv (muhast) glede komunikacije in le-ta občasno ni delovala. Več razlage sledi v poglavju 5, kjer je bolj podrobna razlaga zakaj je prihajalo do teh napak.

Sledilci linij uporabljajo dva svetlobna senzorja za sledenje črni poti – ko se eden od senzorjev premakne s črne črte, robot preprosto zavije v drugo smer. To mu omogoča, da sledi neravnim potem, vse dokler ni notranji polmer zavoja manjši od 100mm (polmer, manjši od 100mm, je bil za robota problematičen). Uporaba dveh svetlobnih senzorjev je bila nemogoča za uporabo z LEGO MindStorms RCX 2.0, saj komplet vključuje le enega. Da bi izkoristil vse RCX module, sem zasnoval še drugo obliko robota, ki ne uporablja nobenih svetlobnih senzorjev (želel sem porabiti vse RCX module), in sicer robota-prenašalca. Robot-prenašalec je statična postaja, ki ima na vrtljivo ploščad nameščen RCX modul, tako da IR senzor preišče področje v krožnem gibanju (kot svetilnik).

Lastnosti robotov:

- Roboti, ki sledijo liniji, imajo dva svetlobna senzorja, ki omogočata robotu, da sledi kateri koli črni liniji, vse dokler je dovolj kontrasta med črno linijo in spodnjo podlago (uporabljen je bil A0 bel papir).
- Robot-prenašalec je statična postaja, ki rotira po vertikalni osi (ta dizajn je bil zasnovan zaradi manjšega števila svetlobnih senzorjev v novih RCX 2.0 Lego kompletih – odločil sem se uporabiti vse RCX module).
- Posebej prilagojena LED LEGO kocka, ki je pritrjena na enega motornih izhodov. LEGO kocka uporablja 10mm LED diode visoke intenzitete, ki zagotavljajo dobro vidnost.
- Roboti imajo zaščito za IR senzorje (kartonasto pokrivalo), da minimizirajo interference fluorescentne svetlobe in naključno razpršene / preusmerjene komunikacije.



Slika 6: Model sledilca linije

4.2. Programska implementacija

Programska oprema, ki omogoča delovanje, je v grobem razdeljena na dva paketa – paket *env*, ki je namenjen delu s strojno platformo in paket *prophet6,* namenjen delovanju zavojnega in PRoPHET protokola.

Vse, povezano s podporo delovanja strojne opreme na posamezni platformi, je realizirano v paketu e*nv*, ki prek javnih vmesnikov daje na voljo vse ključne funkcionalnosti, ki jih na posamezni platformi lahko nadziramo. Javni vmesnik *Machine* tako daje na voljo razrede, ki omogočajo nadzor nad delovanjem naprave – tako imamo možnost, da začnemo s premikanjem, končamo s premikanjem, nadziramo LED diode, sprejmemo zavoj, … Javni vmesnik *Communication* oz. *CommunicationIR* nam daje na voljo razrede, ki omogočajo upravljanje s komunikacijo – tako imamo možnost pošiljanja sporočil, sprejema sporočil in pregleda nad stanjem komunikacije.

Pristop z javnimi vmesniki je bil izbran zaradi razlik med PC in RCX platformo. Na RCX platformi s pomočjo leJOS razredov nadziramo delovanje strojne opreme, medtem ko je na PC platformi posamezne vmesnike potrebno še implementirati. Za primer naj navedemo funkcijo *light*, ki s pomočjo parametra krmili delovanje LED diod. Na RCX platformi funkcija s pomočjo leJOS razreda za nadzor izhodov prižge LED diodo, na PC platformi pa funkcija obarva grafičen gradnik LED diode.

Na drugi strani je v paketu *prophet6* realizirano delovanje zavojnega in PRoPHET protokola. Število šest v imenu je povezano z verzijo implementacije PRoPHET protokola, razvoj je bil iterativen proces, med katerim je bilo potrebno prilagajanje osnovne ideje do delujoče implementacije. Sama implementacija PRoPHET protokola je zasnovana neodvisno od strojne platforme na kateri teče, saj s strojno opremo komunicira prek javnih vmesnikov paketa *env.* To v praksi pomeni, da je celotna implementacija zajeta v razredu *Prophet*.

Zaradi potreb po hkratnem delovanju komunikacije, nadzora gibanja robota in PRoPHET protokola, je programska implementacija izvedena več-nitno, tako da podpora delovanju strojne opreme in PRoPHET protokol tečeta v ločenih nitih.

4.3. PRoPHET

PRoPHET protokol, ki je implementiran v razredu *Prophet*, je razdeljen na tri ključne dele:

1) na jedrno funkcionalnost, ki skrbi za izvajanje vseh ostalih PRoPHET funkcij in krmili napravo,

2) *bundling agent,* ki skrbi za delo z zavoji in

3) *routing agent*, ki skrbi za PRoPHET usmerjanje.

Bundling agent skrbi za delo z zavoji in omogoča ustvarjanje novega zavoja *(createBundle)*, sprejem zavoja *(acceptBundle*) in vračanje zavoja iz shrambe v pomnilniku *(getBundle*) zavoja. Ključnega pomena je funkcija *acceptBundle,* saj skrbi za prepisovanje starejših zavojev, v primeru da začne primanjkovati prostora. Trenutno se prepiše eden izmed shranjenih zavojev brez posebne strategije izbora, lahko pa bi za izbor vpeljali kakšno metriko, ki bi pomagala pri odločanju – na primer FIFO, čas čakanja v shrambi zavojev, pogostost posredovanja zavoja drugim vozliščem, …

Routing agent skrbi za delo s PRoPHET protokolom, kar pomeni, da vsebuje razrede, ki manipulirajo shranjene verjetnosti v PRoPHET usmerjevalni tabeli.

Baza usmerjevalnih informacij je usmerjevalna tabela, ki je zasnovana na verjetnosti, da se bo vozlišče srečalo z določenimi vozlišči. To uporabimo, ko se sprejema odločitev o posredovanju ali neposredovanju sporočil. Ko vozlišče sprejme Bazo usmerjevalnih informacij, posodobi svojo verjetnost, da bo srečalo drugo vozlišče, glede na nove informacije. Baza usmerjevalnih informacij se stara po vnaprej določeni periodi, v skladu s postopki, opisanimi v predhodnem poglavju.

Razredi skrbijo za staranje usmerjevalne tabele, preverjajo pogoje za sprejem posameznega zavoja in skrbijo za izračun sprememb verjetnosti (shranjenih v lastni tabeli) na podlagi prejetih tabel verjetnosti vozlišča, ki jih je poslalo.

4.4. Komunikacija

Komunikacija, ki poteka med vozlišči, temelji na infrardečem prenosu in poteka prek vgrajenih vmesnikov v Lego RCX module ali komunikacijske stolpe. Zaradi lažje uporabe smo se odločili za uporabo starejših komunikacijskih stolpov (ki so del paketa Lego

MindStorms RCX 1.0), ki za povezavo uporabljajo zaporedni vmesnik (novejši uporabljajo USB vmesnik).

Funkcionalnost komunikacije je zajeta v javnih vmesnikih *Communication* in *IRCommunication*, ki skrbita za prejem in pošiljanje podatkov. Podatki se pošiljajo v obliki zavojev, s tem da je maksimalna količina podatkov, ki jo lahko prenesemo v vsakem sporočilu, 37 bajtov. Pri tem so prvi štirje vedno glava sporočila in vključujejo naslednje podatke: tip, zastavica, vir in cilj.

Vsako vozlišče konstantno pošilja *Hello request* sporočila, dokler ne prejme *Hello request* sporočila od drugega vozlišča. Nato zaustavi motorje in preveri katero vozlišče ima nižjo identifikacijo vozlišča – vozlišče samo ali vozlišče, s katerim se povezuje. Za minimalizacijo količine sporočil, ki so istočasno poslana med njima (omejitev v RCX strojni opremi preprečuje dvosmerno (full-duplex) komunikacijo), tisto z nižjo identifikacijo prevzame vlogo 'gospodarja' in zahteva informacijo od drugega, ki prevzame vlogo 'sužnja' in le pošilja odgovore z zahtevanimi informacijami. Ko je vozlišče-gospodar pridobilo vse potrebne informacije, se vlogi vozlišč zamenjata, tako da vozlišče-gospodar postane vozlišče-suženj in obratno.

Izmenjane informacije so:

- Baza usmerjevalnih informacij (usmerjevalna tabela)
- *Bundle offer* (ponudba zavojev, ki so na voljo za posredovanje)
- morda tudi *Bundle requests* (zahteva po prenosu zavojev, ki jih lahko posreduje).

Bundle requests so izmenjani v primeru, ko obstaja zavoj, ki izpolnjuje pogoj GRTR strategije posredovanja.

Ko obe vozlišči s sporočanjem zaključita, izključita komunikacijo, da se ne bi takoj spet srečali, vključita motorje in se odpeljeta stran.

4.5. Demonstracija delovanja

Vsak Lego RCX modul omogoča razvijalcu določene predstavitvene zmožnosti. Obstaja možnost uporabe vgrajenega LCD zaslona za predstavitev podatkov (celo kratkih nizov), uporabe notranjega zvočnika za proizvajanje zvoka ter LED diod (ki se namestijo na enega izmed perifernih vmesnikov in so vključene v Lego MindStorms RCX paket) za dodatne

vizualne povratne informacije. Izkazalo se je, da majhen LCD zaslon ni uporaben za prave predstavitvene namene, saj iz večjih razdalj ni viden. Zato sem se odločil, da pritrdim LED diode, toda tiste, ki so prišle v originalnem paketu, so bile premajhne in preslabo vidne.

Na svetovnem spletu sem našel rešitev – priprava posebnih LEGO kock, v katere sem vgradil dve LED diode visoke intenzivnosti (rdečo in modro). LED diodi sta vgrajeni v standardno 6x2 LEGO kocko in vezani tako, da se prižigata izmenično – odvisno od smeri toka. Zaradi take vezave je bilo možno krmiliti LED diodo kar s pomočjo razreda za upravljanje z motorjem – vrtenje motorja naprej je prižgalo modro, vrtenje nazaj pa rdečo LED diodo.

V sam Java razred, namenjen krmiljenju robota, sem dodal nekaj dodatnih razredov, kar je omogočilo uporabo modre in rdeče LED diode za predstavitev podatkovne poti, kot tudi za poročanje o delovanju vozlišča. Za dodaten efekt je uporabljen tudi zvočnik.

Ključno vprašanje pri demonstraciji delovanja je pregled poti, ki jo je ubral določen zavoj pri potovanju skozi omrežje. Mehanika PRoPHET protokola naj bi omogočila samo smotrno pošiljanje zavojev, torej samo tistim vozliščem, ki bodo lahko zavoj predali prejemniku ali vsaj vozlišču, ki je prejemniku bližje. Za namen prikaza te mehanike sem sestavil poseben zavoj, PRoPHET ping, ki ni nosil nobenih pomembnih podatkov, ampak je bil namenjen označevanju poti. Vsaka točka, ki je takšen zavoj prejela, je prižgala eno izmed LED diod.

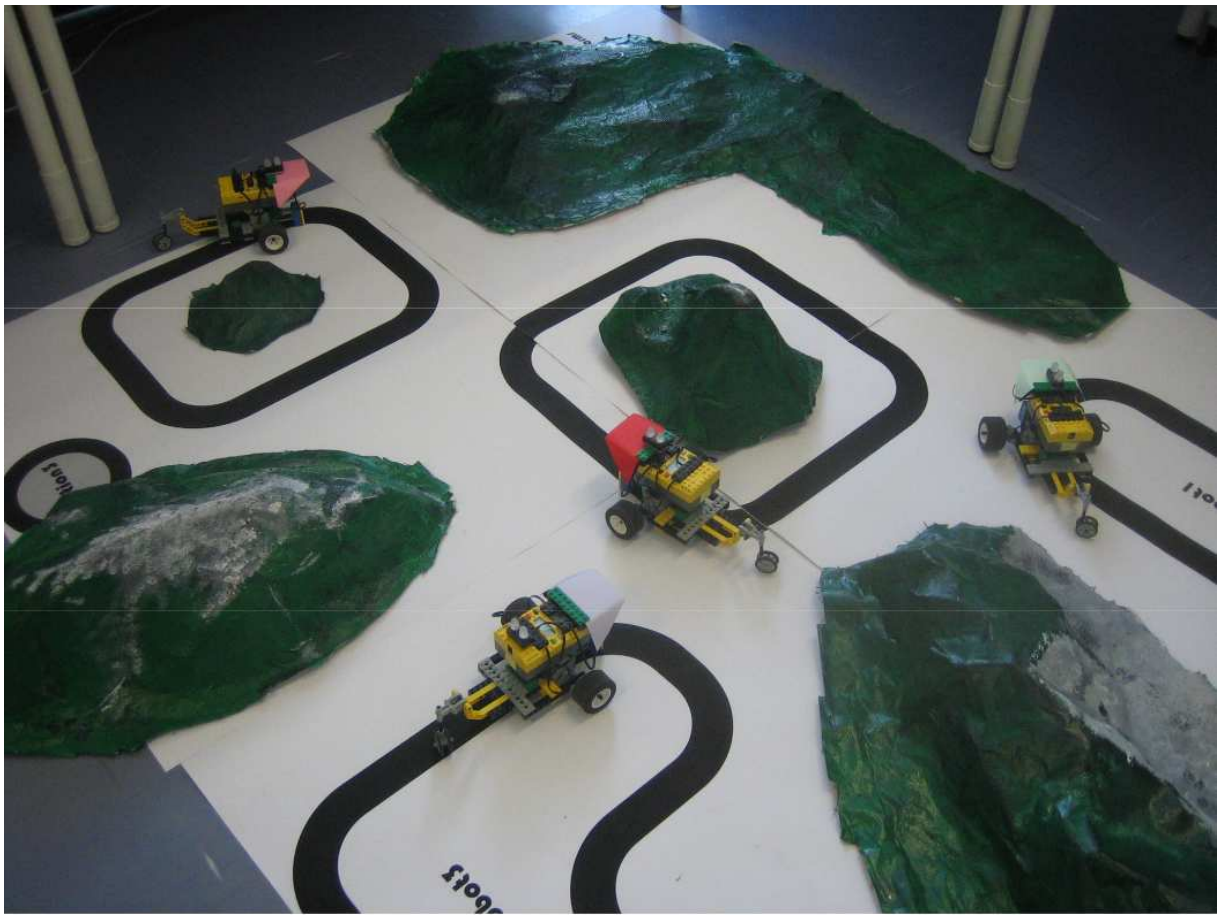
Poročanje o delovanju omrežja je potekalo na naslednji način:

1. Na osebnem računalniku sem v PRoPHET aplikaciji ustvaril PRoPHET ping paket z drugim osebnim računalnikom kot naslovnikom. Izmenjava paketa med osebnima računalnikoma naj bi potekala s pomočjo Lego MindStorms robotov, ki se gibljejo po narisanih poteh.

2. Ko imata dva robota svoja IR senzorja v vidnem polju drug drugega, se lahko prične komunikacija. Ob začetku komunikacije sem prižgal eno izmed LED diod, kar je opozorilo opazovalca, da poteka izmenjava podatkov. LED diodo je prižgal gospodar povezave in jo ob zamenjavi vloge ugasnil.

3. V primeru da se je med robotoma uspešno izvedla izmenjava vsaj enega izmed shranjenih zavojev, je na to opozoril s piskom vgrajenega zvočnika. Če je zavoj vseboval PRoPHET ping, potem je prejemnik prižgal izbrano LED diodo, kar je omogočilo sledenje poti, ki jo je zavoj opravil pri potovanju prek omrežja.

4. Ko je sporočilo uspešno dostavljeno končnemu prejemniku (drugi osebni računalnik), le-ta odgovori na PRoPHET ping paket, ki se pošlje nazaj v omrežje in ugasne prižgane označevalne LED diode.



Slika 7: Mobilna RCX vozlišča med demonstracijo delovanja

4.6. Grafično orodje za interakcijo in pregled delovanja

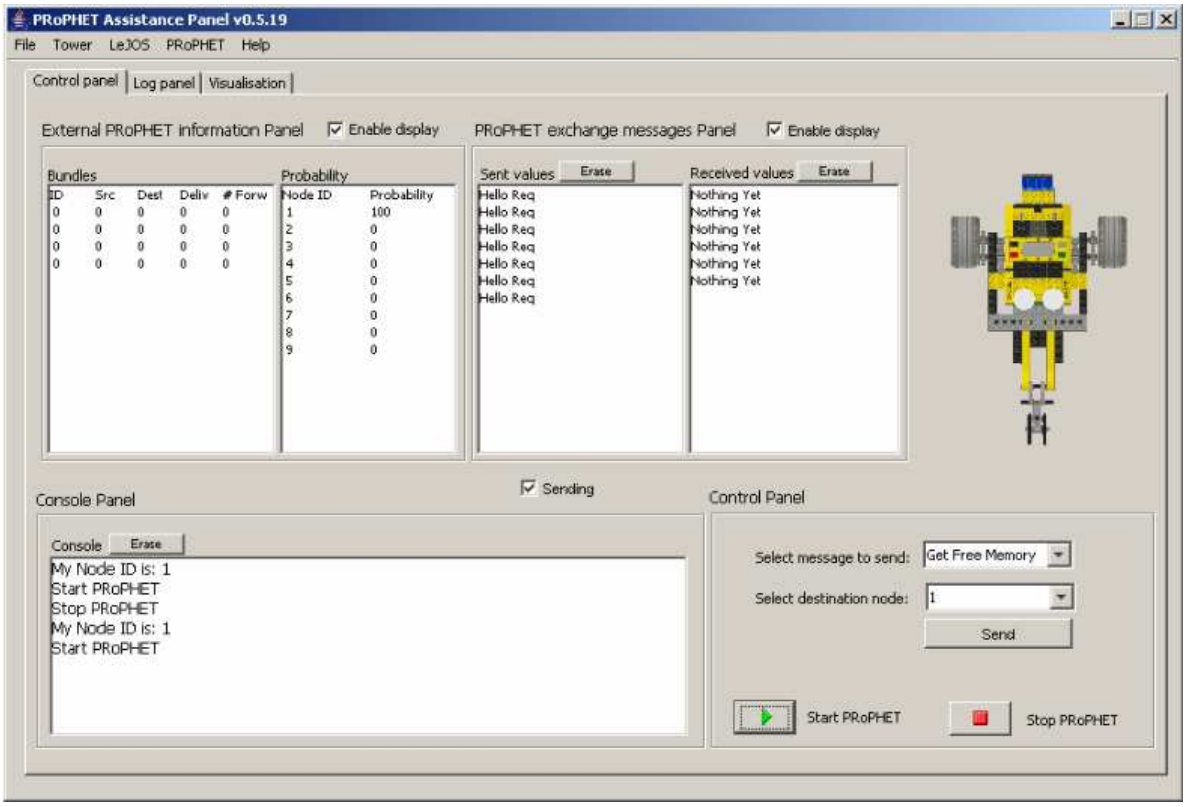
Sama postavitev grafičnega vmesnika je bila pripravljena s pomočjo orodij, vgrajenih v Borland JBuilder, kar je močno olajšalo delo, saj omogoča pripravo Java Swing vmesnikov z uporabo vgrajenega grafičnega orodja.

Grafično orodje za interakcijo je razdeljeno na dva glavna dela, ki se nahajata vsak v enem zavihku. Prvi – *control panel* je namenjen nadzoru delovanja naprave in pošiljanju ter sprejemanju sporočil. Drugi – *log panel* – je namenjen zajemu in pregledovanju zgodovine dogodkov, ki je shranjena na vsaki izmed naprav.

Control panel

Uporabnika ob zagonu programa pričaka odprt zavihek *Control panel*, ki mu omogoča pregled stanja in daje možnost za upravljanje PRoPHET vozlišča, ki teče v sklopu *PRoPHET Assistance Panela*. *External PRoPHET information panel* ponuja uporabniku nadzor nad shranjenimi zavoji ter tabelo dostavnih verjetnosti. *PRoPHET exchange messages panel* omogoča pregled nad poslanimi in prejetimi sporočili, desno od le-tega se nahaja slika robota, na katerem je razvidna aktivnost LED diod.

V aplikaciji so že pripravljeni testni paketi, ki jih lahko pošljemo drugemu vozlišču. Uporabnik si enostavno izbere tip paketa in ga s pritiskom na tipko send pošlje naslovniku.

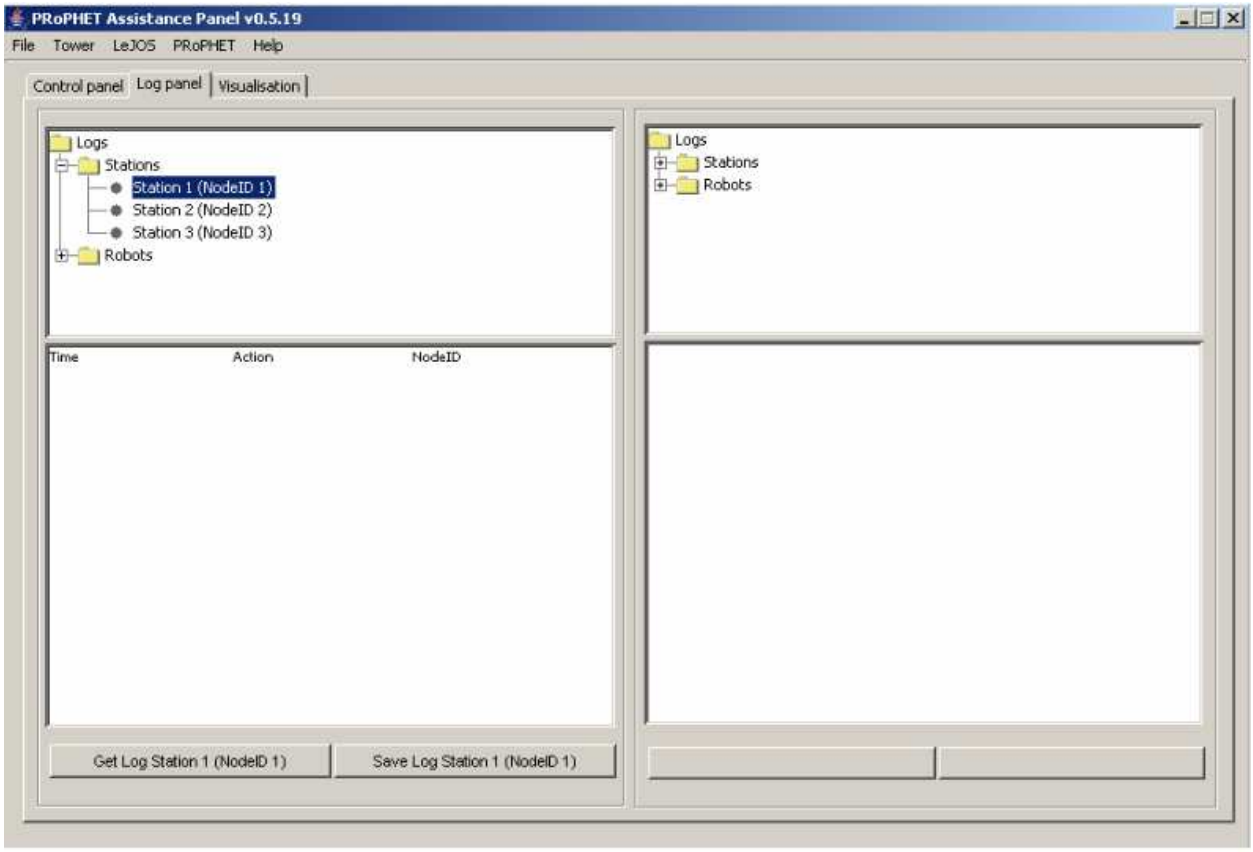


Slika 8: Grafično orodje PRopHET Assistance Panel s pogledom na Control panel

Log panel

V *log panel* zavihku ima uporabnik možnost shranjevanja in pregledovanja zgodovine dogodkov. Sam zavihek je razdeljen na dva enaka dela, kar omogoča uporabniku primerjavo zgodovine dogodkov med dvema vozliščema. Tako ima možnost analize zgodovine dogodkov, ki mu lahko pove ali dve napravi pri medsebojni komunikaciji delujeta pravilno. V zgornjem delu polovice se nahaja seznam vseh naprav, v spodnjem delu pa okno z vpogledom v zgodovino izbrane naprave.

Postopek uporabe: Uporabnik si na seznamu izbere vozlišče, s katerega bi rad prenesel zgodovino dogodkov. Vozlišče postavi v doseg IR stolpa in sproži prenos z naprave s pomočjo pritiska na gumb *Get Log*. Zgodovina dogodkov se prikaže v spodnjem oknu. Če želi primerjati dve napravi med seboj, potem postopek enostavno ponovi in si s pomočjo seznamov nastavi katero napravo naj program prikazuje na levi in katero na desni strani.



Slika 9: Grafično orodje PRopHET Assistance Panel s pogledom na Log panel

5. TEŽAVE

5.1. RCX modul

RCX modul je daleč od tega, da bi bil popoln in večino težav povzročajo IR vrata, ki jih uporabljamo za komunikacijo z zunanjim svetom. IR vrata uporabljamo za programiranje in komunikacijo med RCX moduli. Prvo očitno težavo je predstavljala fluorescentna svetloba, ki je ovirala IR komunikacijo – v svojem laboratoriju nisem mogel programirati RCX-a brez uporabe posebnih 'ohišij za programiranje' (kartonastih škatel), ki so omogočile programiranje v popolni temi.

Ob preizkušanju komunikacije med RCX moduli so se pojavile težave pri izmenjavi podatkov; vir je bila ponovno fluorescentna svetloba. Težavo je odpravila uporaba posebne zaščite za IR senzorje, kartonastega pokrivala, ki je preprečevalo direkten sij fluorescentne svetlobe iz stropnih svetil. Tako se je IR komunikacija sicer izboljšala, vendar ni bila povsem predvidljiva – še vedno je bila potrebna določena mera sreče za delujoče IR povezave, še posebej z RCX 2.0 moduli.

USB IR stolp, ki je vključen v RCX 2.0, je povzročal veliko težav. Obstajajo trije različni načini delovanja in vsi so premočni ter se odbijajo od zidov (potrebno je omeniti, da so vsa RCX vozlišča bila nameščena tako, da so uporabljala nizko moč IR, saj so se drugače žarki odbijali od zidov in predmetov ter onemogočali komunikacijo). Prisiljen sem bil uporabljati stolpe s priključkom za zaporedni vmesnik (te sem dobil v starih LEGO MindStorms paketih).

5.2. Java in leJOS

Med pripravo implementacije PRoPHET protokola, ki deluje na RCX modulih v okolju leJOS, sem naletel na ogromno težav zaradi strojnih omejitev same platforme. Pomnilnik na RCX modulih je izjemno omejen (okoli 20KB, od tega jih več kot 6KB potrebuje že samo osnoven leJOS), kar je otežilo pripravo Java razredov.

Namesto izjemno pregledne implementacije sem moral večino razredov ponovno napisati in prilagoditi za delo na RCX modulih. Odstranil sem vse nepotrebne funkcije, ki so zavzemale prostor, potreben za uspešno izvajanje. Tako, na primer, nisem smel uporabiti vgrajenega znakovnega LCD prikazovalnika, saj so knjižnice za izris na zaslon preobsežne.

Dodatno težavo predstavlja tudi dejstvo, da leJOS nima vgrajenega čistilca pomnilnika (ang. *garbage collector*), kar seveda pomeni, da ni bilo mogoče uporabljati nobenih dinamičnih podatkovnih struktur in so bile vse statično določene še pred prvim zagonom.

Vsem tem potezam navkljub mi je uspelo pridobiti zgolj dovolj pomnilnika za zagon rešitve in shranjevanje podatkov 4 zavojev ter 176 vnosov v dnevnik dogodkov.

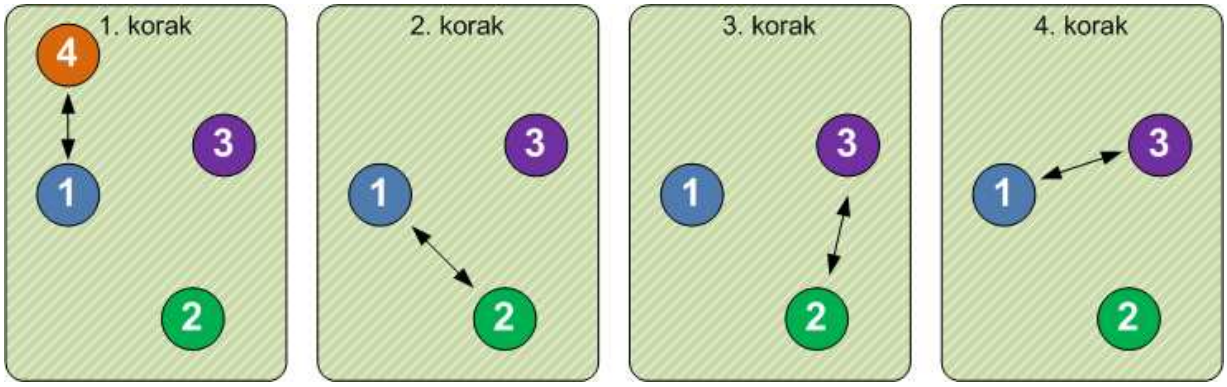
5.3. PRoPHET

Pred začetkom dela je bila dokumentacija o PRoPHET usmerjevalnem protokolu še na zelo nizkem nivoju, saj do moje implementacije še ni obstajala implementacija, na kateri bi lahko bolj podrobno preveril delovanje protokola. Tako so v sami dokumentaciji dokaj arbitrarno določili primerne konstante PRoPHET enačb. Med testiranjem delovanja implementacije so se začele pojavljati težave pri dostavi paketov.

Problem enačb je namreč v dejstvu, da morajo biti konstante prilagojene za razmere, v katerih se bo PRoPHET protokol uporabljal. Če sem preveč spremenil testni scenarij – namesto dveh mobilnih vozlišč sem jih uporabil šest – se je zgodilo, da se zaradi prenizke konstante β enostavno ni prenesla verjetnost, da ima določeno mobilno vozlišče posredno pot do naslovnika sporočila. Ko sem konstanto β popravil, se je zgodilo obratno – pri spremembi testnega scenarija se je zgodilo, da so zaradi prehodnosti med vozlišči vse vrednosti v tabelah dostavnih verjetnosti sčasoma konvergirale k 1, kar je zaradi GRTR strategije posredovanja onemogočilo dostavo paketov.

Težava je najbolj izražena v omrežjih, kjer se vozlišča srečujejo ciklično. V takem primeru se zaradi lastnosti posredne povezljivosti zgodi, da povečujemo dostavno verjetnost tudi tistim vozliščem, ki jih že dolgo nismo srečali.

Primer težave, ki jo povzroča enačba za izračun dostavne verjetnosti zaradi posredne povezljivosti:



Slika 10: Težave zaradi posredne povezljivosti

Vozlišče 1 se v prvem koraku sreča z vozliščem 4 in shrani dostavno verjetnost v lastno tabelo dostavnih verjetnosti. V drugem koraku vozlišči 1 in 2 izmenjata tabeli dostavnih verjetnosti in vozlišče 2 vnese posredno dostavno verjetnost vozlišču 4. V tretjem koraku se srečata 2 in 3, kjer se postopek ponovi in vozlišče 3 vnese posredno dostavno verjetnost za vozlišče 4. Ko se srečata 1 in 3, se prenese posredna dostavna verjetnost za vozlišče 4, čeprav vozlišča 4 nobeden že dolgo časa ni videl.

Da se težava zaradi posredne povezljivosti izrazi, zadostujeta že dve vozlišči, ki se srečujeta dovolj pogosto. Medsebojna izmenjava lahko zaradi posrednih dostavnih verjetnosti povzroči konvergenco vseh vrednosti, ki jih imata shranjeni v tabelah, proti 1. Težava se pojavi, ker to negativno vpliva na celotno omrežje, saj bosta obe vozlišči to napako prek izmenjave dostavnih verjetnosti posredovala še vsem ostalim vozliščem, kar bo, seveda, vplivalo na dostavo zavojev.

Enaka težava se zgodi tudi če se naenkrat sreča zelo veliko točk, ki med seboj ustvarijo povezave, kar spet privede do situacije, kjer dostavne verjetnosti konvergirajo proti 1. Ta problem je poimenovan "težava polnega parkirišča" – zaradi razmisleka kaj se zgodi, ko se zaradi nekega dogodka na majhnem področju zbere veliko vozlišč; na primer, vsi prebivalci gorskih vasi na nekem skupnem festivalu v dolini.

6. Zaključek

Splošna razširjenost računalniških omrežij je povzročila, da jih praktično vsak prebivalec razvitega sveta vsakodnevno uporablja pri delu, študiju ali zabavi. Zato morda niti ne vidimo potrebe po drugačnih tipih omrežij, ki za svoje delovanje ne potrebujejo posebne infrastrukture, saj je komunikacijska infrastruktura prisotna povsod.

Izkaže se, da je to prepričanje zmotno in da obstaja mnogo okolij, v katerih klasična omrežja, ki temeljijo na uporabi obstoječih protokolov, niso najbolj primerna. Odložljiva omrežja, z uporabo dodatnega nivoja med aplikativnim in omrežnim nivojem, omogočajo uporabo računalniških aplikacij tudi tam, kjer prej zaradi omejitev okolja ali opreme le-to ni bilo mogoče. Z uporabo odložljivih omrežij omogočamo nastanek novih senzorskih omrežij z nižjo porabo energije in dostavo šolskih materialov v nerazvita okolja.

Projekt sem začel z jasnim ciljem; pripraviti zanimivo predstavitev delovanja odložljivih omrežij in PRoPHET protokola, ki bo razumljiva tudi osebi, ki nima posebnega strokovnega znanja o omreženosti. Mislim, da sem zastavljeni cilj uspešno dosegel, saj lahko delovanje enostavno prikažemo z razvitimi rešitvami. Lego RCX moduli, ki se premikajo po prostoru in si med seboj izmenjujejo informacije, so dobro orodje za prikaz širjenja zavojev skozi omrežje, sama aplikacija z grafičnim vmesnikom pa uporabniku uspešno omogoča pregled delovanja in interakcijo z omrežjem.

Med delom sem naletel na kar nekaj težav – ugotovil sem, da je res težko nastaviti konstante, ki so del enačb PRoPHET protokola, na stopnje, ki bi bile primerne fizičnemu okolju, v katerem naj bi se uporabljal. Izkazalo se je, da je PRoPHET izjemno občutljiv na nastavitve konstant in da je edini primeren način za določanje primernih vrednosti preizkus v simulatorju ali v praksi (če je to mogoče).

Tudi delo z izbrano platformo ni bilo najbolj preprosto, saj je platforma glede strojnih zmogljivosti izjemno omejena (omejen pomnilnik) in muhasta (infrardeč vmesnik). Njihovi muhasti naravi navkljub mislim, da je bil izbor Lego RCX modulov pravilna odločitev, saj lahko zelo nazorno pokažejo delovanje omrežja.

Kljub težavam se je izkazalo, da je PRoPHET primeren protokol za usmerjanje podatkov v odložljivih okoljih. Izsledki iz tega projekta so bili pomembni za nadaljevanje razvoja v sklopu Sami Network Connectivity, v sklopu katerega so – tudi s pomočjo mojih ugotovitev

– v letu 2007 na področju severne Švedske uspešno postavili delujoče odložljivo omrežje, ki za usmerjanje zavojev uporablja PRoPHET usmerjevalni protokol.

7. Seznam slik in enačb

Seznam slik:

Seznam enačb:

8. Reference

[1] BrickOS. (2009) Dostopno na: http://brickos.sourceforge.net/.

[2] Grašič S. (2008). Implementacija protokola PRoPHET v odložljivih omrežjih. *Diplomsko delo*. Ljubljana: Fakulteta za računalništvo in informatiko.

[3] IETF (2009) Overview of the IETF. Dostopno na: http://www.ietf.org/overview.html.

[4] LEGO Mindstorms. (2009) Dostopno na: http://www.lego.com/eng/education/mindstorms/home.asp?pagename=rcx.

[5] LeJOS - Java for LEGO Mindstorms. (2009) Dostopno na: http://lejos.sourceforge.net/rcx.php

[6] Lindgren A., Doria A., Davies E., Grasic S. (2009) Probalistic Routing Protocol for Intermittently Connected Networks. Dostopno na: http://www.ietf.org/id/draft-irtf-dtnrg-prophet-02.txt

[7] Lindgren A., Doria A., and Schelén O. (2003) "Probabilistic Routing in Intermittently Connected Networks", *SIGMOBILE Mobile Computing Comm. Rev.*, vol. 7.

[8] pbForth Weblog Home. (2009) Dostopno na: http://www.hempeldesigngroup.com/lego/pbForth/index.html

[9] Postel, J. (1981) "Transmission Control Protocol", RFC 793.

[10] The Sámi Network Connectivity Project . (2009) Dostopno na: http://www.snc.sapmi.net.

[11] Wizzy Digital Courier. (2009) Dostopno na: http://www.wizzy.org.za

[12] Waitzman, D. (2008) RFC 1149: Standard for the transmission of IP datagrams on avian carriers. Dostopno na: http://www.faqs.org/rfcs/rfc1149.html

[13] Zhang, P., et al. (2004) "Hardware Design Experiences in ZebraNet". *Proc. ACM Conference on Embedded Networked Sensor Systems*, Baltimore.