# Seminarska naloga

pri predmetu **Modeliranje in simulacije**

Fakulteta za računalništvo in informatiko
Univerza v Ljubljani

Kranj, 26. 07. 2006

avtorja:
Tine Lesjak,
Borut Lesjak

mentor:
izr. prof. dr. Nikolaj Zimic

# Vsebina

# Opis problema

Mobilne naprave pošiljajo GPS podatke o svoji geografski lokaciji (v nadaljevanju sporočila) na strežnik prek računalniškega omrežja (mobilno omrežje – GPRS, Internet, LAN). Vsaka naprava ima nastavljeno točno določeno frekvenco pošiljanja sporočil (npr. na 5 sekund), vendar sporočila zaradi značilnosti mobilnih omrežij (moč signala, napake, preobremenjenost, potovanje signala) ne zmorejo vedno prispeti na cilj v tem času.

Radi bi opazovali, kakšna je torej dejanska frekvenca pošiljanja sporočil mobilnih naprav glede na njihovo nastavljeno frekvenco. Pri tem nas zanima povprečni pretečeni čas (v nadaljevanju **povprečna zakasnitev**) in največji pretečeni čas (v nadaljevanju **največja zakasnitev**) med dvema sporočiloma.

# Opis rešitve

Mobilne naprave pošiljajo sporočila vsakih 5 sekund. Pri tem se z motnjami ustvarijo napake pri pošiljanju. Naprava mora zato po nekem času sporočilo poslati še enkrat. S tem simuliramo zakasnitev, ki jo dejansko ustvarijo slab mobilni signal, ki je odvisen od položaja naprave v prostoru, ali preobremenjenost omrežja (zasedenost GPRS celic v bazni postaji).

Strežnik na drugi strani sprejema sporočila in vodi statistiko za vsako napravo posebej o povprečni zakasnitvi in največji zakasnitvi.

# Predpostavke in poenostavitve

Kakovost mobilnega signala je posledica gibanja mobilnih naprav v prostoru. Prostor ni nikoli povsem pokrit z močnim signalom. Ponekod je signal slab pri čemer prihaja do napak pri prenosu. Zgodi se lahko tudi, da je omrežje za kratek čas preobremenjeno in tako ni dovolj GPRS celic na voljo za uspešen prenos sporočila. Pri tem nastajajo krajše zakasnitve. Če predpostavimo, da napake pri prenosu ustvarjajo zakasnitev, potem lahko simulacijo poenostavimo tako, da z eno logično komponento za vsako napravo ustvarjamo samo napake.

Za dandanašnje razmere mobilne naprave pošiljajo zelo malo podatkov. Zato je ves vmesni mehanizem (GSM bazne postaje, pretvorniki, gatewayi, usmerjevalniki, Internet, telefonske centrale in ne nazadnje prepustnost vodnikov) za dostavo sporočila od bazne postaje pa do strežnika zanemarljiv in vpliva samo na absolutno zakasnitev dostave enega sporočila (latency), ne pa na pretečeni čas med dvema sporočiloma.

V praksi imamo v večjem prostoru (recimo v Sloveniji) večje število baznih postaj. Vendar če se omejimo na manjši prostor in predpostavimo, da ima bazna postaja veliko prepustnost prenosa podatkov, lahko v simulaciji uporabimo le eno bazno postajo.

Mobilne naprave se v praksi gibljejo po prostoru zelo počasi glede na hitrosti prenosa podatkov. Zato lahko poenostavimo, da se napravi ne spremeni moč signala vsak trenutek, temveč le na vsakih nekaj sekund. Neuspešno poslana sporočila se v napravi kopičijo, dokler se moč signala ne izboljša.

# Natančen opis

## Mobilna naprava

Vsaka mobilna naprava ima svojo TCP aplikacijo in svojo logično komponento za ustvarjanje napak. Logične komponente in TCP aplikacije posameznih naprav so med seboj neodvisne.

Aplikacija vsakih 5 sekund pošlje sporočilo strežniku.
Logična komponenta neodvisno od pošiljanja sporočila nastavi verjetnost, da bo poslani bit napačen (bitna napaka) na komunikacijskem kanalu med napravo in bazno postajo. Verjetnost bitne napake se izračuna po naslednjem algoritmu:

1. Naključno se izračuna položaj mobilne naprave v prostoru.

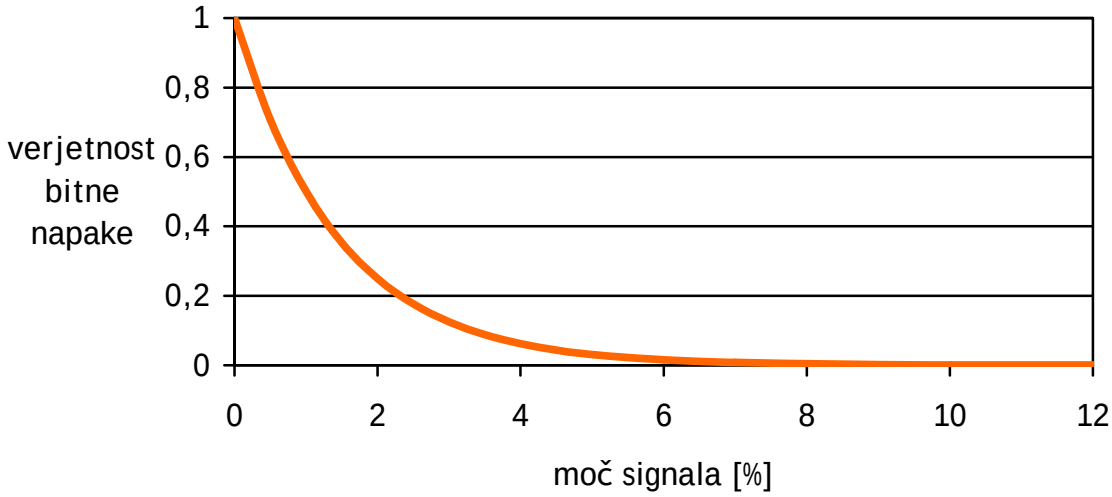
```
lokacija = intrand(100) + 1
```

2. Moč mobilnega signala se izračuna glede na položaj naprave:
   - če se mobilna naprava nahaja na lokaciji, kjer je moč signala polna, se takšna moč tudi vzame (100 %),
   - če se mobilna naprava nahaja na lokaciji, kjer ni signala, se vzame nična moč (0 %),
   - če se mobilna naprava nahaja na lokaciji, kjer moč signala ni polna ali nična, se naključno izračuna moč signala, ki je lahko na intervalu od 1-99 %.

```
močSignala =   100 %, če lokacija <= "fullSignalQuality"
               0 %, če lokacija > "fullSignalQuality" in <=
               ("fullSignalQuality" + "noSignalQuality")
               intrand(99) + 1, sicer
```

3. Izračuna se verjetnost bitne napake.
   Verjetnost bitne napake lahko zaseda vrednosti od 1 (napačen je vsak bit) do parametra "najmanjša verjetnost bitne napake" (minError), po formuli:

$$napaka = 2^{(-močSignala - "minError")} + "minError"$$




4. Izračuna se trajanje nahajanja mobilne naprave v tem področju.

```
trajanje =      "areaTimeMin" + intrand("areaTimeMax" -
                "areaTimeMin " + 1)
```

*Opombi za formule:*

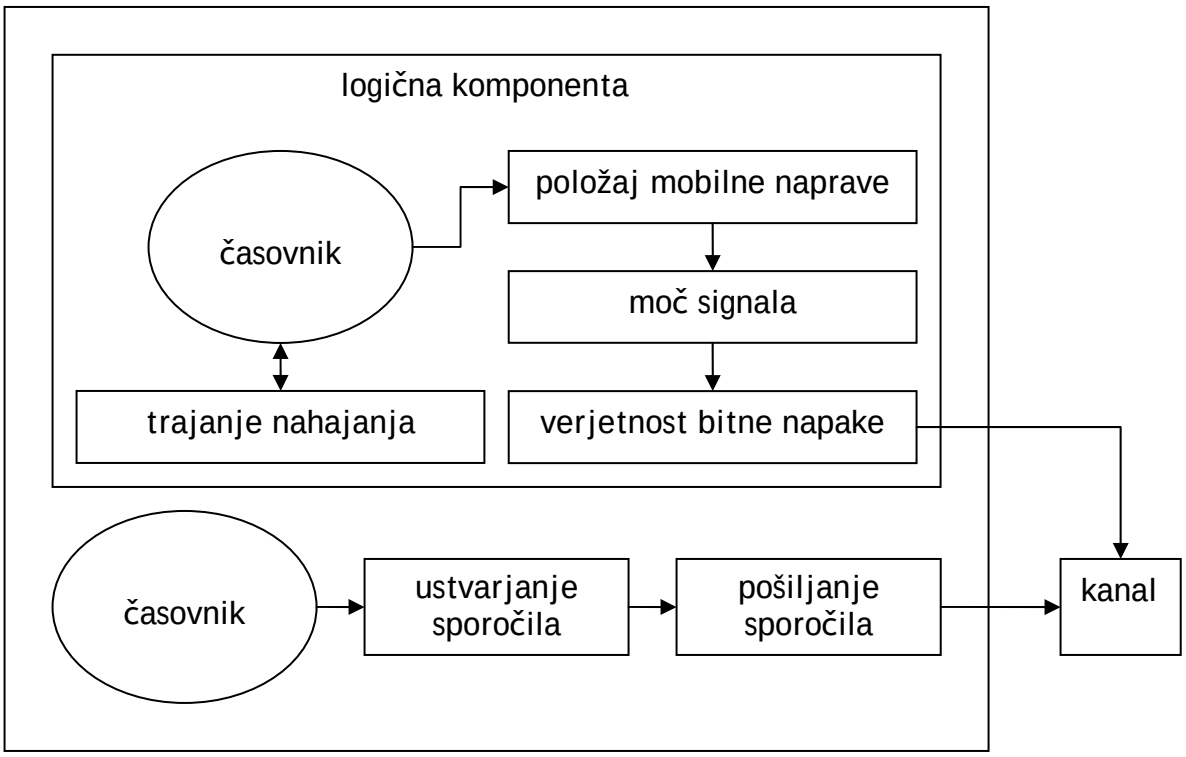
- *`intrand` je naključni generator celih števil na intervalu od 0 do (vrednost v oklepaju)-1,*
- *parametri v narekovajih so opisani v razdelku Parametri.*

Tako dobljena verjetnost bitne napake se poda komunikacijskemu kanalu, ki samodejno ustvarja bitne napake pri prenosu podatkov.

Aplikacija nato ustvari sporočilo (brezvsebinsko), dolgo povprečno 63 znakov, in ga pošlje nižjim nivojem, da le-ti poskrbijo za prenos sporočila do strežnika.

Če pride pri prenosu do napake, to bazna postaja tudi ugotovi (sporočilo je posebej označeno) in paket enostavno izbriše. Mobilna naprava nato samodejno po določenem času (retransmission timeout [20] - najmanj 1 sekundo) spet pošlje isto sporočilo. To se samodejno dogaja na nižjem nivoju od aplikativnega (datalink nivo OSI modela [13]).

Verjetnost bitne napake na komunikacijskem kanalu logična komponenta spreminja časovno neodvisno od regularnega pošiljanja sporočila (ima svoj časovnik).



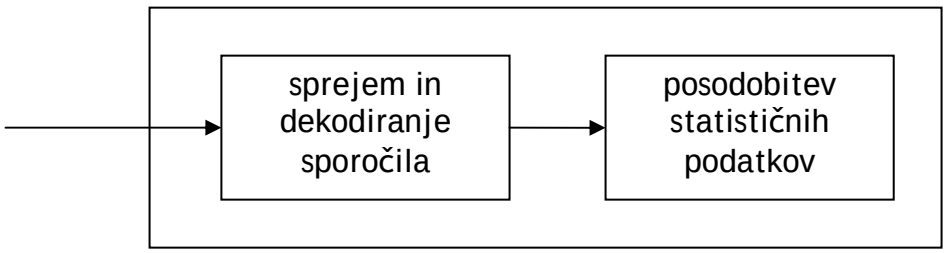

Abstraktni model aplikacije v mobilni napravi

## Strežnik

Na strežniku teče TCP aplikacija, ki vodi statistiko za vsako mobilno napravo posebej. Ob priklopu naprave na strežnik aplikacija ustvari nov vektor podatkov, v katerega si shranjuje statistiko za to napravo. Strežnik nato čaka na prvo sporočilo. Statistiko začne voditi po prejetju prvega sporočila.
Pri prejetju sporočila se izračuna zakasnitev od prejetja prejšnjega sporočila:

- zakasnitev se doda v povprečno zakasnitev,
- če je zakasnitev večja od največje zakasnitve, se zabeleži kot največja.

Aplikacija v strežniku gleda neodvisno na zakasnitev od frekvence pošiljanja v mobilni napravi. Strežnik napravi ne pošilja ničesar.

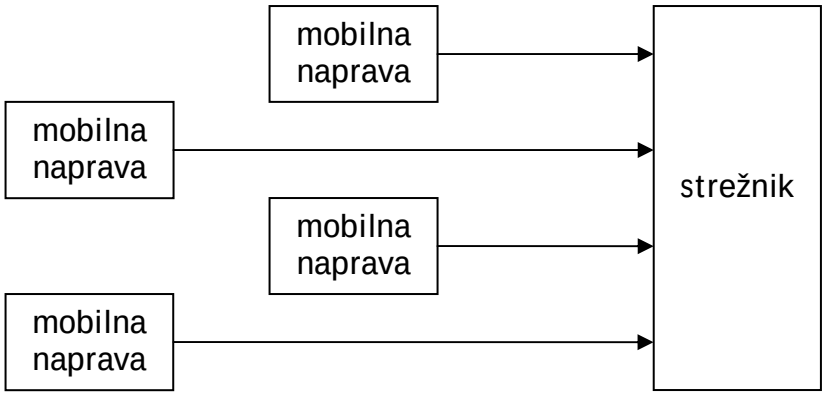



Abstraktni model aplikacije v strežniku

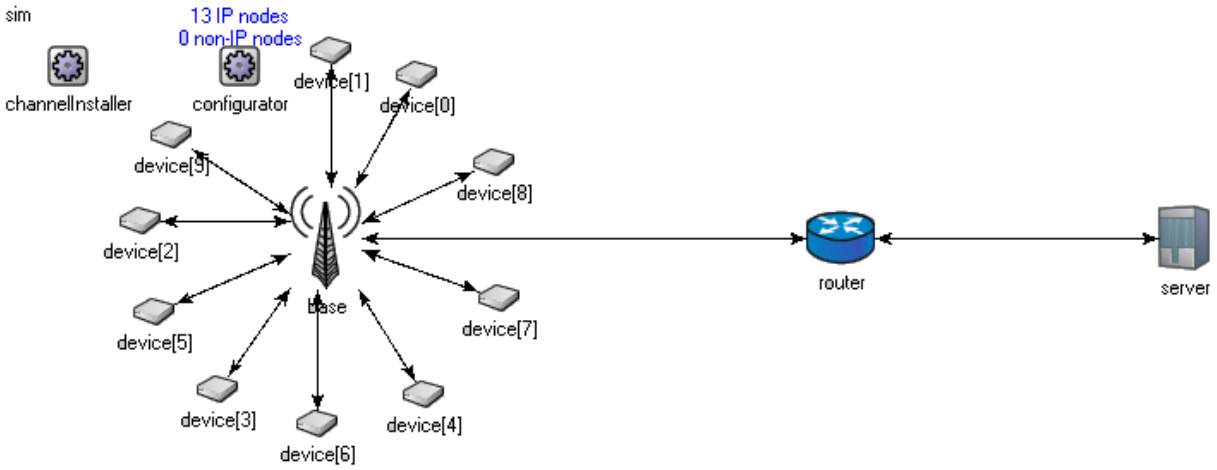
# Načrt simulacije

## Zgradba

Simulacija je glede na poenostavitve preprosta, kar zadeva števila komponent. Sestavljena je iz večjega števila mobilnih naprav, ki so prek PPP (Point-to-Point Protocol [12]) povezane na eno bazno postajo. Bazna postaja je usmerjevalnik, ki podatke pošilja po "internetni" liniji (PPP) do usmerjevalnika pred strežnikom. Ta usmerjevalnik je lokalne narave, zato je strežnik nanj priklopljen na LAN način (Ethernet [15]). Strežnik je en.




Abstraktni model simulacije




Slika simulacije

## Orodje

Za simulacijo uporabljava OMNeT++ [2] diskretno dogodkovno simulacijsko orodje. OMNeT++ je program, ki je podlaga za gradnjo in pogon simulacij. Vsebuje jedro (kernel), ki vodi simulacijski čas in ustrezno generira in razpošilja dogodke simulacijskim komponentam (modulom). Program pa vsebuje tudi programske knjižnice in dokumentacijo za gradnjo novih simulacij. Sprogramiran je v C++ programskem jeziku in podpira samo module napisane v tem jeziku. Da pa ni potrebno čisto vsega napisati v C++ programskem jeziku, OMNeT++ podpira številne druge, bolj primerne formate. Module, medmodulske povezave in opis sporočil lahko kar opisno sprogramirano v lastnem NED jeziku. Parametre lahko trdno zakodiramo, jih zapišemo v posebni konfiguracijski datoteki ali jih vnesemo pri zagonu simulacije.

INET [4] je ogrodje, ki vsebuje vse potrebne računalniške omrežne module ter ima implementirane vse nivoje OSI modela [13]. INET je v bistvu samo paket številčnih modulov z izvorno kodo, ki ga namestimo poleg OMNeT++ in prevedemo. Svojo simulacijo, ki vsebuje INET module, zgradimo tako, da preprosto dodajamo svoje module v ta paket in vse skupaj prevajamo.

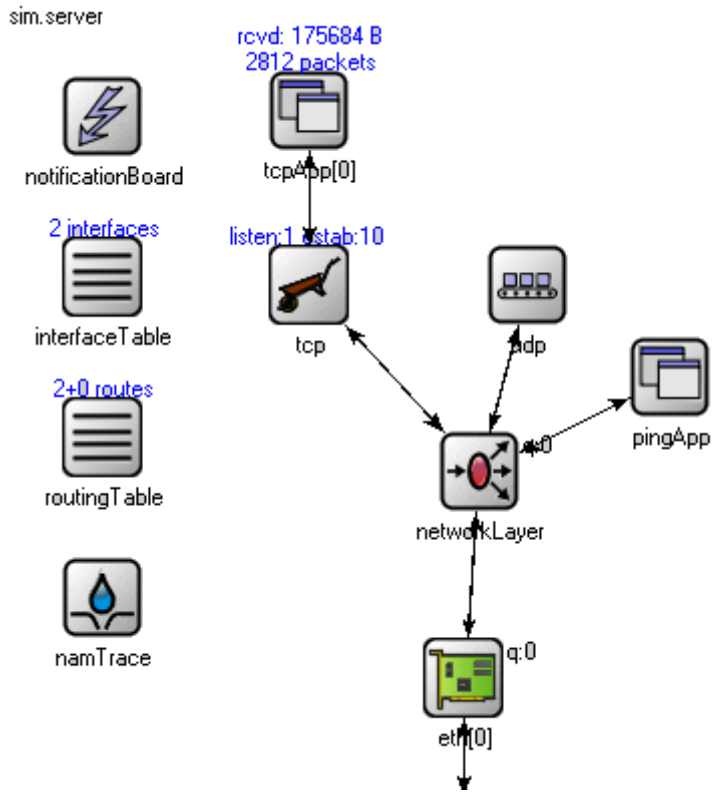
## Izvorna koda

Zgradbo simulacije (topologijo) predstavlja glavni sestavljen modul (`Misim.ned`), ki opisuje vse enostavne module in kanale (povezovalne linije med moduli) ter povezuje module med sabo.

Vsebuje:

- opis internetne linije
- opis GPRS linije
- povezave med mobilnimi napravami, bazno postajo, lokalnim usmerjevalnikom in strežnikom

Mobilna naprava, bazna postaja, lokalni usmerjevalnik in strežnik so zgrajeni iz več že implementiranih INET omrežnih modulov. Mobilna naprava in strežnik imata na koncu verige modulov TCP aplikaciji, ki sva ju za simulacijo sama implementirala.




Slika strežnika z vsemi nivoji

Za mobilno napravo sva napisala **TCPMobileDeviceApp** modul, ki skrbi za vzpostavitev povezave do strežnika ter vso potrebno logiko za frekvenčno pošiljanje sporočil in nastavljanje bitne napake. Zmodeliran je s tremi datotekami:

- `TCPMobileDeviceApp.ned` je topologija TCP aplikacije kot modula
- `TCPMobileDeviceApp.h` je opis izvorne kode modula (C++ header datoteka)
- `TCPMobileDeviceApp.cc` je izvorna koda modula (C++ source datoteka)

Za strežnik sva napisala **TCPServerStatApp** modul, ki skrbi za sprejem sporočil, prepoznavo mobilnih naprav in posodabljanje statistike. Modul vsebuje tudi objekt, ki skrbi za vektorje, kamor se zapisuje statistika za vsako napravo posebej.
Zmodeliran je s tremi datotekami:

- `TCPServerStatApp.ned`
- `TCPServerStatApp.h`
- `TCPServerStatApp.cc`

Za zagon simulacije in parametre skrbijo konfiguracijske datoteke. Glavna je `omnetpp.ini`, ki vključuje tudi konfiguracijske datoteke za vsako komponento posebej. Tako je za mobilne naprave datoteka `devices.ini`, za bazno postajo `base.ini`, za lokalni usmerjevalnik `router.ini` in za strežnik `server.ini`. Parametri, ki so ključni za simulacijo, so opisani v naslednjem razdelku.

# Parametri

## Število mobilnih naprav

*vnesemo ob zagonu simulacije*

Število mobilnih naprav povezanih na bazno postajo. Pri večjem številu naprav dobimo bolj raznolike rezultate simulacije. Število praktično ne vpliva na obremenjenost omrežja, saj naprave pošiljajo zelo malo podatkov.

## Dolžina sporočila

*devices.ini -> **.device[*].tcpApp[0].numBytes*

Dolžina sporočila, ki ga pošiljajo mobilne naprave je določeno z normalno porazdelitvijo s srednjo vrednostjo 63 in standardno deviacijo 2, kar daje povprečno 63 znakov (bajtov). Ta številka izvira iz narave ti. GPRMC stavkov (standardizirani NMEA stavki [11]), ki se uporabljajo pri sporočanju geografskega položaja v GPS-u.

## Zagon naprave

*devices.ini -> **.device[*].tcpApp[0].startTime*

Čas, ki ga porabi mobilna naprava, da se zažene (da začne pošiljati sporočila). V praksi se naprave zelo različno prebujajo (od 10 sekund do nekaj minut), zato sva ta parameter določila z normalno porazdelitvijo s srednjo vrednostjo 40 in standardno deviacijo 10.

## Frekvenca pošiljanja sporočil

*devices.ini -> **.device[*].tcpApp[0].idleTime*

Frekvenca pošiljanja sporočil določa, na koliko časa mobilna naprava pošlje eno sporočilo strežniku. V praksi se za največjo frekvenco uporablja 5 sekund, za najmanjšo pa tudi nad 1 minuto. V simulaciji vedno uporabljava 5 sekund.

## Velikost popolnega področja

*devices.ini -> **.device[*].tcpApp[0].fullSignalPercent*

Popolno področje je prostor, ki je pokrit s 100 % močnim mobilnim signalom. Velikost tega področja je podana v odstotkih glede na ves prostor. Ta parameter uporablja logična komponenta za računanje moči signala.

### Velikost nepokritega področja

*devices.ini -> **.device[*].tcpApp[0].noSignalPercent*

Nepokrito področje je prostor, kamor mobilni signal ne seže (njegova moč je 0 %). Velikost tega področja je podana v odstotkih glede na ves prostor. Ta parameter uporablja logična komponenta za računanje moči signala.

### Najmanjša verjetnost bitne napake

*devices.ini -> **.device[*].tcpApp[0].minError*

Najmanjša verjetnost bitne napake je verjetnost bitne napake na GPRS liniji, ko je moč signala 100 % (v popolnem področju). Manjša ne more biti. Primerna verjetnost bitne napake je $10^{-6}$.

### Trajanje nahajanja v istem področju

*devices.ini -> **.device[*].tcpApp[0].areaTimeMin*
*devices.ini -> **.device[*].tcpApp[0].areaTimeMax*

Trajanje nahajanja v istem področju pove, koliko časa se položaj mobilne naprave ne bo spremenil (in s tem tudi moč signala). Parametra določata interval od najkrajšega do najdaljšega trajanja, ki ju uporablja logična komponenta za računanje časa za naslednji premik naprave.

### Prepustnost, zakasnitev in verjetnost bitne napake internetne linije

*Misim.ned -> channel internetLine*

Prepustnost internetne linije je hitrost prenosa podatkov med bazno postajo in lokalnim usmerjevalnikom pred strežnikom. Ta je na primer 10 Mb/s. Zakasnitev je nastavljena na 1 ms. Verjetnost bitne napake je nastavljena na $10^{-6}$. Ti trije parametri praktično ne vplivajo na simulacijo, saj mobilne naprave pošiljajo zelo malo podatkov.

### Prepustnost in zakasnitev GPRS linije

*Misim.ned -> channel gprsUp*
*Misim.ned -> channel gprsDown*

Prepustnost GPRS linije je hitrost prenosa podatkov med posamezno mobilno napravo in bazno postajo. Ta znaša največ 24 kb/s za smer od naprav (upload) in 36 kb/s za smer do naprav (download) (GPRS Class 10 - Two Up [16]). Odzivnost (latency) pri GPRS povezavi je tipično 600-700 ms [8]. Vrednost zakasnitve GPRS linije sva nastavila z normalno porazdelitvijo s srednjo vrednostjo 325 in standardno deviacijo 20, kar predstavlja povprečno zakasnitev v eno smer 325 ms.

### Prepustnost LAN linije

```
router.ini -> **.router.eth[*].mac.txrate
server.ini -> **.server.eth[*].mac.txrate
```

Prepustnost LAN linije je hitrost prenosa podatkov med lokalnim usmerjevalnikom in strežnikom. Pri večini današnjih LAN povezav se podatki prenašajo s hitrostjo 100 Mb/s v duplex načinu.

# Rezultati

## Razlaga rezultatov

Pri pregledu statistike, ki jo vodi strežnik, se da razbrati, da povprečno sporočila prispejo v enakem času kot so oddana s strani naprave, ne glede na napake in zakasnitve. Ta rezultat je presenetljiv, vendar hkrati potrjuje odlično zasnovo današnjih računalniških omrežij.

*Kako je lahko povprečna zakasnitev enaka frekvenci pošiljanja?*
Sporočila, ki niso bila uspešno poslana se kopičijo v napravi (samodejno na nižjih nivojih), dokler naprava ne preide na področje z dovolj dobrim signalom. Takrat se vsa neposlana sporočila pošljejo praktično v hipu. Strežnik zabeleži le veliko zakasnitev prvega sporočila, vsa ostala sporočila pa imajo nično zakasnitev, saj pripotujejo hkrati s prvim. Matematično povprečje se zato izravna nazaj.

Lahko pa se zgodi, da se mobilna naprava prevečkrat nahaja v področju s slabim signalom (prevečkrat je hotela ponovno poslati). Tedaj naprava povezavo enostavno prekine in se začne ponovno vzpostavljati povezave s strežnikom. Pri tem se izgubijo vsa sporočila, ki so se nakopičila. Ker je število prispelih sporočil na strežniku manjše od števila sporočil, ki jih je hotela poslati naprava, je povprečna zakasnitev večja od frekvence pošiljanja.

Največja časovna zakasnitev med dvema sporočiloma od naprave do naprave zelo variira. Vsaka naprava ima namreč svojo logično komponento, ki krmili verjetnost bitne napake. S tem simuliramo povsem neodvisno gibanje naprav v prostoru, kar daje različne rezultate.
Opaziti je, da so največje časovne zakasnitve bistveno večje, kot bi pričakovali glede na verjetnost bitne napake in dolžino sporočil.

*Zakaj so največje zakasnitve tako velike?*
TCP protokol uporablja tako imenovan Nagleov algoritem [18], ki zmanjša število paketov, ki morajo biti poslani prek omrežja, in s tem izboljša učinkovitost omrežij. Deluje tako, da pošiljatelj zbira kratka sporočila, ki jih kasneje pošlje v enem paketu. Zbira jih samo takrat, kadar ne dobi potrdila od sprejemnika. Če se preselimo na naš primer, ugotovimo, da mobilna naprava začne zbirati sporočila takoj, ko za prvo poslano sporočilo ne dobi potrditve od strežnika (vzrok je slab mobilni signal). Kasneje bi naprava lahko poslala vsa zbrana sporočila, ker je verjetno že prešla na področje z boljšim mobilnim signalom, vendar jih zbira tako dolgo, dokler velikost vseh zbranih sporočil postane tako velika, da se jih izplača poslati. Glede na to, da je velikost enega sporočila dokaj majhna, traja precej časa, da se jih nakopiči dovolj.
Dodatno k tako velikim zakasnitvam pripomore tudi čas za ponovno pošiljanje nepotrjenega sporočila (retransmission timeout [20]). Vsakokrat, ko naprava ponovno ne dobi potrditve, se retransmission timeout poveča za faktor 2.

## Meritve

Pri vseh meritvah sva simulirala 10 mobilnih naprav en dan simulacijskega časa (86.400 sekund). Pri tem parametre "dolžina sporočila", "zagon naprave" in "najmanjša verjetnost bitne napake" nisva spreminjala. Nastavljeni so bili na:

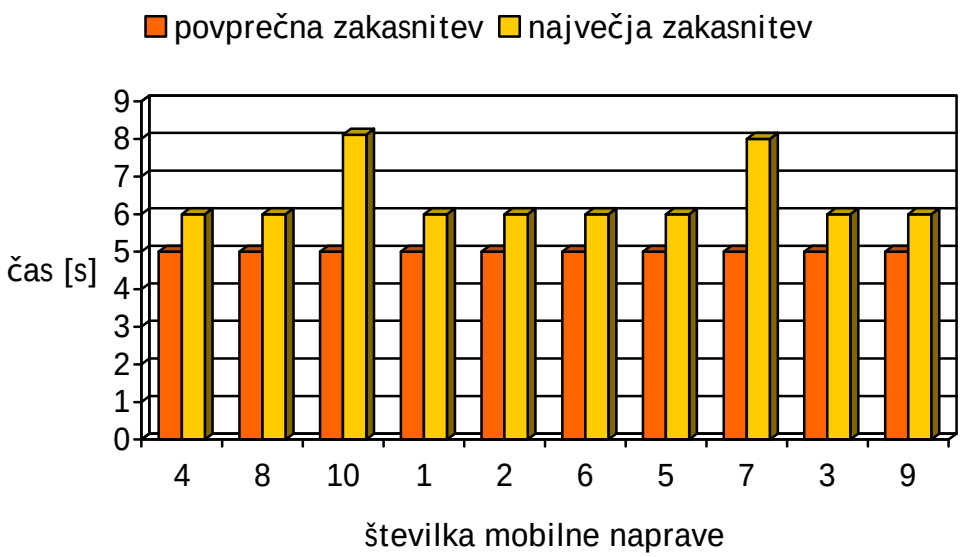
```
**.device[*].tcpApp[0].numBytes=truncnormal(63, 2)
**.device[*].tcpApp[0].startTime=truncnormal(40, 10)
**.device[*].tcpApp[0].minError=0.000001
```

### Meritev št. 1: Potrditev modela

Meritev za potrditev modela z vedno polnim signalom:

```
**.device[*].tcpApp[0].idleTime=5s
**.device[*].tcpApp[0].fullSignalPercent=100
**.device[*].tcpApp[0].noSignalPercent=0
**.device[*].tcpApp[0].areaTimeMin=5s
**.device[*].tcpApp[0].areaTimeMax=5s
```

| številka mobilne naprave | število prekinitev | število prejetih sporočil | povprečna zakasnitev [s] | največja zakasnitev [s] |
|---:|---:|---:|---:|---:|
| 4 | 0 | 17.274 | 5,00000 | 6,00201 |
| 8 | 0 | 17.273 | 5,00000 | 6,00100 |
| 10 | 0 | 17.273 | 5,00000 | 8,11341 |
| 1 | 0 | 17.273 | 5,00000 | 6,00167 |
| 2 | 0 | 17.272 | 5,00000 | 6,00167 |
| 6 | 0 | 17.272 | 5,00000 | 6,00234 |
| 5 | 0 | 17.272 | 5,00000 | 6,00201 |
| 7 | 0 | 17.270 | 5,00000 | 8,00000 |
| 3 | 0 | 17.270 | 5,00000 | 6,00301 |
| 9 | 0 | 17.269 | 5,00000 | 6,00167 |
| **povprečje** | **0** | **17.272** | **5,00000** | **6,41288** |




Največja zakasnitev med dvema sporočiloma je povprečno trajala **128 %** povprečne zakasnitve.

Glede na to, da je pri tej meritvi signal vedno poln, bi se moralo vsako sporočilo takoj prenesti do strežnika. Vendar temu ni tako, kakor kažejo tudi rezultati, saj se napake vseeno dogodijo. GPRS linija med mobilno napravo in bazno postajo in internetna linija med bazno postajo in lokalnim usmerjevalnikom imata najmanjšo verjetnost bitne napake nastavljeno na $10^{-6}$, kar je očitno dovolj, da do napake pride vsaj enkrat na vseh desetih napravah (retransmission timeout [20] je 1 sekunda), na dveh celo dvakrat zaporedoma (retransmission timeout se drugič poveča na 2 sekundi, skupaj 3 sekunde).
Model je pravilen, saj pri polnem signalu nič drugega ne vpliva na napake.

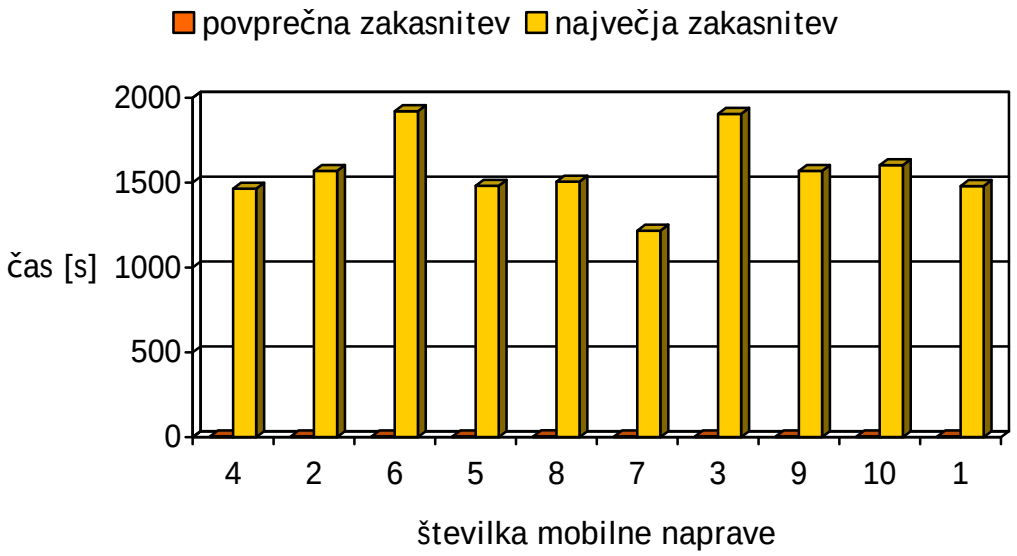
## Meritev št. 2: Skrajne razmere

Meritev s skrajno nemogočimi razmerami, ko moč mobilnega signala lahko zaseda samo med sabo enako verjetni dve vrednosti: 100 % ali 0 %.

```
**.device[*].tcpApp[0].idleTime=5s
**.device[*].tcpApp[0].fullSignalPercent=50
**.device[*].tcpApp[0].noSignalPercent=50
**.device[*].tcpApp[0].areaTimeMin=5s
**.device[*].tcpApp[0].areaTimeMax=5s
```

| številka mobilne naprave | število prekinitev | število prejetih sporočil | povprečna zakasnitev [s] | največja zakasnitev [s] |
|---:|---:|---:|---:|---:|
| 4 | 1 | 16.969 | 5,08619 | 1464,65 |
| 2 | 1 | 16.968 | 5,08966 | 1569,11 |

| | | | | |
|---|---|---|---|---|
| 6 | 2 | 16.689 | 5,17458 | 1920,99 |
| 5 | 2 | 16.664 | 5,17654 | 1480,84 |
| 8 | 4 | 16.090 | 5,36696 | 1505,95 |
| 7 | 0 | 17.268 | 5,00005 | 1217,33 |
| 3 | 0 | 17.257 | 5,00181 | 1904,77 |
| 9 | 3 | 16.469 | 5,24267 | 1568,98 |
| 10 | 3 | 16.280 | 5,27728 | 1602,65 |
| 1 | 4 | 16.092 | 5,36418 | 1470,74 |
| **povprečje** | **2** | **16.675** | **5,17799** | **1570,60** |




Največja zakasnitev med dvema sporočiloma je povprečno trajala **303 krat** več od povprečne zakasnitve.

Rezultati kažejo zelo velike največje zakasnitve, ki so posledica že omenjenega Nagleovega algoritma [18] in tega, da je praktično vsako drugo sporočilo napačno poslano. Moč signala ne alternira pravilno med 100 % in 0 %, saj je lahko tudi večkrat zaporedoma 0 %. Dodatno še k velikosti največje zakasnitve vpliva tudi čas za ponovno pošiljanje nepotrjenega sporočila (retransmission timeout [20]), ki se ob vsakem izteku poveča za faktor 2.

## Meritev št. 3: Poseljeno okolje

Pri tej meritvi s parametri nekako upoštevava dejansko stanje v Sloveniji. Statistika [17] pravi, da je pozidanega 2,8 % slovenskega ozemlja, ceste pa zasedajo 1 % ozemlja. Upava si trditi, da je popolno področje veliko vsaj 5 % ozemlja. Mobitel d.d. zatrjuje, da je pokritost prebivalstva 99,2 % [22]. Če prištejeva še nekaj desetink odstotka cest, kjer so predori in signala ni, sva parameter nepokritega področja nastavila na 1 %. Poleg teh dveh parametrov sva nastavila tudi trajanje nahajanja

mobilne naprave na istem področju, tako da zaseda bolj realne vrednosti od 3 sekund do 10 minut.

```
**.device[*].tcpApp[0].idleTime=5s
**.device[*].tcpApp[0].fullSignalPercent=5
**.device[*].tcpApp[0].noSignalPercent=1
**.device[*].tcpApp[0].areaTimeMin=3s
**.device[*].tcpApp[0].areaTimeMax=10m
```

| številka mobilne naprave | število prekinitev | število prejetih sporočil | povprečna zakasnitev [s] | največja zakasnitev [s] |
|---:|---:|---:|---:|---:|
| 4 | 0 | 17.274 | 5,00000 | 980,000 |
| 8 | 0 | 17.273 | 5,00000 | 1220,000 |
| 10 | 0 | 17.266 | 5,00000 | 912,952 |
| 1 | 0 | 17.273 | 5,00000 | 979,999 |
| 2 | 0 | 17.233 | 5,00000 | 1229,980 |
| 6 | 0 | 17.272 | 5,00000 | 1215,000 |
| 5 | 0 | 17.272 | 5,00000 | 740,000 |
| 7 | 0 | 17.270 | 5,00000 | 980,000 |
| 9 | 0 | 17.269 | 5,00000 | 740,001 |
| 3 | 6 | 17.179 | 5,00000 | 979,999 |
| **povprečje** | **0,6** | **17.258** | **5,00000** | **997,793** |


Največja zakasnitev med dvema sporočiloma je povprečno trajala **200 krat** več od povprečne zakasnitve.

Podroben pogled v potek simulacije odkrije, da se je mobilna naprava št. 3 na začetku kar 500 sekund nahajala v področju s 4 % močnim signalom, zato se je uspešno povezala na strežnik šele v sedmem poizkusu.

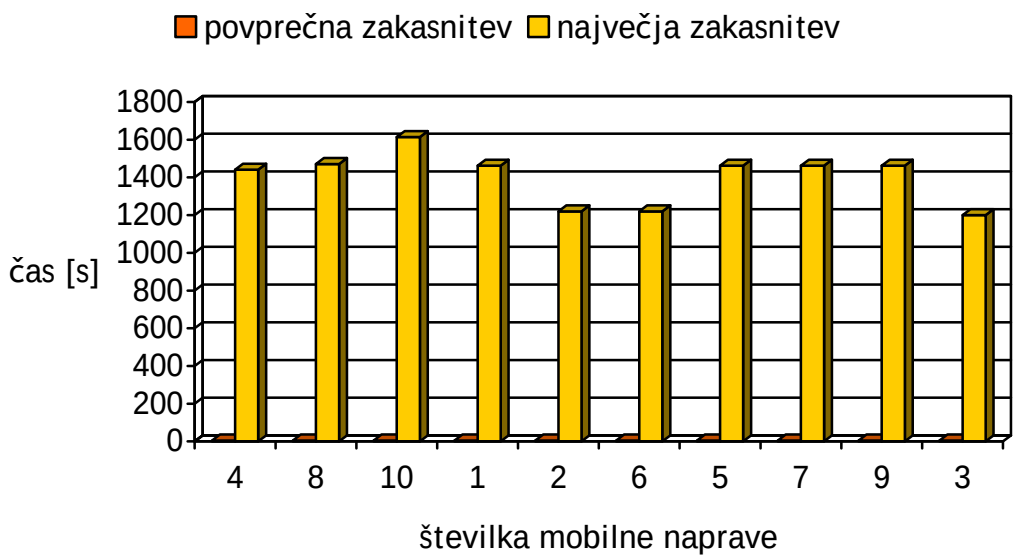
Glede na to, da je povprečna zakasnitev popolnoma enaka frekvenci pošiljanja sporočil, lahko zatrdiva, da je model popolnoma normalen in da odraža resnično stanje.

## Meritev št. 4: Manj poseljeno okolje

Pri tej meritvi sva parametre nastavila tako, da simulacija odraža manj poseljeno okolje, kjer je gostota baznih postaj manjša:

```
**.device[*].tcpApp[0].idleTime=5s
**.device[*].tcpApp[0].fullSignalPercent=3
**.device[*].tcpApp[0].noSignalPercent=10
**.device[*].tcpApp[0].areaTimeMin=3s
**.device[*].tcpApp[0].areaTimeMax=10m
```

| številka mobilne naprave | število prekinitev | število prejetih sporočil | povprečna zakasnitev [s] | največja zakasnitev [s] |
|---:|---:|---:|---:|---:|
| 4 | 0 | 17.274 | 5,00000 | 1440,98 |
| 8 | 1 | 16.982 | 5,08578 | 1470,64 |
| 10 | 3 | 16.951 | 5,09491 | 1613,74 |
| 1 | 1 | 16.981 | 5,08580 | 1461,74 |
| 2 | 0 | 17.272 | 5,00000 | 1220,00 |
| 6 | 0 | 17.272 | 5,00000 | 1220,00 |
| 5 | 2 | 16.689 | 5,17458 | 1461,73 |
| 7 | 1 | 16.774 | 5,08684 | 1461,66 |
| 9 | 1 | 16.977 | 5,08580 | 1461,65 |
| 3 | 6 | 17.179 | 5,00000 | 1200,69 |
| **povprečje** | **1,5** | **17.035** | **5,06137** | **1401,28** |



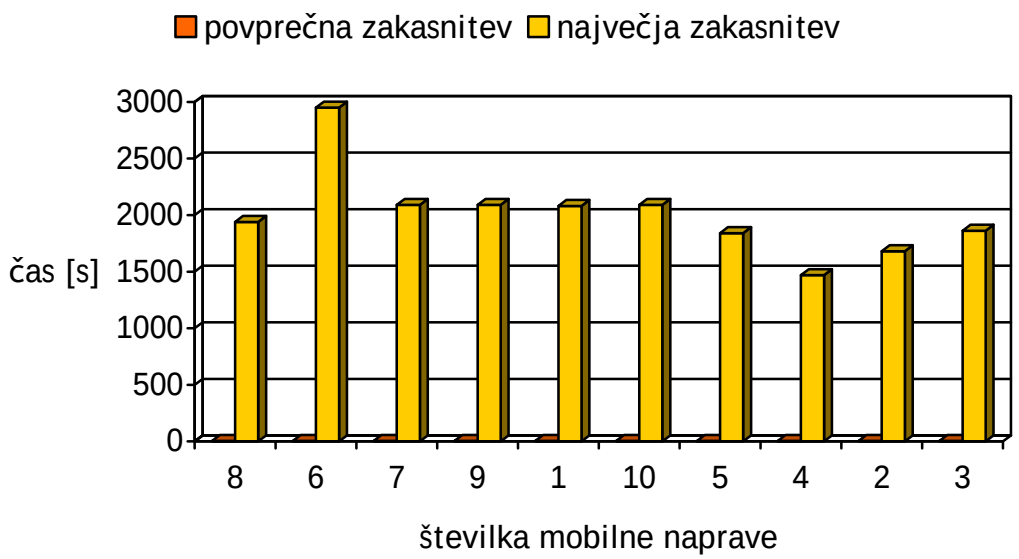

Največja zakasnitev med dvema sporočiloma je povprečno trajala **277 krat** več od povprečne zakasnitve.

## Meritev št. 5: Neposeljeno okolje

Pri tej meritvi sva parametre nastavila tako, da simulacija odraža praktično neposeljeno okolje, kjer so bazne postaje, na primer, le v večjih mestih:

```
**.device[*].tcpApp[0].idleTime=5s
**.device[*].tcpApp[0].fullSignalPercent=1
**.device[*].tcpApp[0].noSignalPercent=30
**.device[*].tcpApp[0].areaTimeMin=3s
**.device[*].tcpApp[0].areaTimeMax=10m
```

| številka mobilne naprave | število prekinitev | število prejetih sporočil | povprečna zakasnitev [s] | največja zakasnitev [s] |
|---:|---:|---:|---:|---:|
| 8 | 13 | 16.232 | 5,23358 | 1938,66 |
| 6 | 28 | 16.000 | 5,39852 | 2950,67 |
| 7 | 17 | 15.347 | 5,62673 | 2090,66 |
| 9 | 20 | 15.689 | 5,45599 | 2090,65 |
| 1 | 23 | 15.210 | 5,66772 | 2081,74 |
| 10 | 14 | 15.563 | 5,50903 | 2090,74 |
| 5 | 18 | 15.599 | 5,50395 | 1841,73 |
| 4 | 25 | 15.984 | 5,33654 | 2470,65 |
| 2 | 16 | 16.395 | 5,17790 | 1681,11 |
| 3 | 32 | 15.653 | 5,43244 | 1862,66 |
| **povprečje** | **20,6** | **15.767** | **5,43424** | **2109,93** |




Največja zakasnitev med dvema sporočiloma je povprečno trajala **388 krat** več od povprečne zakasnitve.

# Viri


[1] N. Zimic, Predloge s predavanj. MIS.pdf. 2006.
http://lrss.fri.uni-lj.si/sl/teaching/mis/

[2] OMNeT++. OMNeT++ User Manual. 2006.
http://www.omnetpp.org/doc/manual/usman.html

[3] OMNeT++. OMNeT++/OMNEST Simulation Library. 2006.
http://www.omnetpp.org/doc/api/index.html

[4] OMNeT++. INET Framework. 2006.
http://www.omnetpp.org/pmwiki/index.php?n=Main.INETFramework

[5] OMNeT++. INET Framework for OMNeT++/OMNEST, Release 2006-03-30.
http://www.omnetpp.org/doc/INET/neddoc/index.html

[6] Wikipedia. Global System for Mobile Communications. 20:42, 2 July 2006.
http://en.wikipedia.org/wiki/Gsm

[7] G. Tabuashvili. Manager of Radio Frequencies of MagtiCom. 2006.
http://www.magtigsm.com/magazine/2003-2/2003-2-7.html

[8] Wikipedia. General Packet Radio Service. 06:55, 27 June 2006.
http://en.wikipedia.org/wiki/GPRS

[9] Wikipedia. GPRS Core Network. 16:47, 3 June 2006.
http://en.wikipedia.org/wiki/GPRS_Core_Network

[10] J. Narikka. Wireless Developer Network. GPRS Performance Issues For Wireless Application Designers. 2006.
http://www.wirelessdevnet.com/channels/wireless/features/gprsperformance.html

[11] D. DePriest. NMEA data. 2006.
http://www.gpsinformation.org/dale/nmea.htm

[12] Network Working Group. The Point-to-Point Protocol (PPP). July 1994.
http://tools.ietf.org/html/1661

[13] Wikipedia. OSI model. 03:10, 6 July 2006.
http://en.wikipedia.org/wiki/OSI_model

[14] Wikipedia. Ethernet. 23:42, 5 July 2006.
http://en.wikipedia.org/wiki/Ethernet

[15] IEEE. IEEE 802.3 CSMA/CD (ETHERNET). 19 Jun 06.
http://www.ieee802.org/3/

[16] GSM World. GPRS Class Type. 2006.
http://www.gsmworld.com/technology/gprs/class.shtml

[17] Statistični urad Republike Slovenije. Slovenija v številkah. 2005.
http://www.stat.si/pub_slovenija.asp

[18] Wikipedia. Nagle's algorithm. 06:24, 5 July 2006.
http://en.wikipedia.org/wiki/Nagle's_Algorithm

[19] TechTarget. SearchNetworking.com Definitions. Nagle's algorithm. 30 Aug 2003.
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci754347,00.html

[20] Network Working Group. RFC 2988 - Computing TCP's Retransmission Timer. November 2000.
http://www.faqs.org/rfcs/rfc2988.html

[21] Mobitel d.d. Osnovni podatki omrežja Mobitel GSM/UMTS. 2006.
http://www.mobitel.si/slo/Ponudba/GSMnarocniki/OMobitelGSM/Osnovnipodatki/default.asp