Elektrotehniški vestnik 74(4): 189-194, 2007
Electrotechnical Review: Ljubljana, Slovenija

Izmenjava in shranjevanje simulacijskih podatkov v telekomunikacijah s programskim orodjem CostGlue

Dragan Savić, Sašo Tomažič, Janez Bešter in Matevž Pustišek

Fakulteta za elektrotehniko, Tržaška 25, 1000 Ljubljana, Slovenija
E-pošta: dragan.savic@fe.uni-lj.si

Povzetek. Izmenjava in shranjevanje simulacijskih podatkov sta pomembni opravili pri izvajanju simulacij. Količina simulacijskih podatkov je lahko dokaj velika, obenem pa so lahko podatki v različnih formatih. Pretvorba velike količine podatkov med različnimi formati je lahko izredno zamudno opravilo. V tem članku se osredotočamo na simulacije v telekomunikacijah. Tako smo preučili potrebe telekomunikacijskega področja ter definirali referenčni model simulacijskega procesa. Skladno s potrebami smo razvili programsko orodje CostGlue, ki je centralni repozitorij za podatke, ustvarjene z različnimi simulacijskimi orodji, in pretvornik različnih izhodnih formatov v različne vhodne formate. CostGlue ima modularno arhitekturo programske opreme. S tem pušča odprte možnosti za nadaljnji razvoj in prispevke drugih raziskovalnih področij. Jedro programskega orodja je aplikacija CoreGlue in skrbi za povezavo s skladovnico podatkov. CoreGlue je enoten vmesnik za vpis podatkov v skladovnico in branje podatkov iz nje. Specifične funkcije, kot so uvoz in izvoz podatkov ter različni matematični izračuni, so izvedene kot množica samoopisljivih modulov, ki se nalagajo po potrebi. Grafični vmesnik je spletni brskalnik, kar uporabniku omogoča prijazen in učinkovitejši oddaljeni dostop do aplikacije. Programski paket CostGlue bomo izdali v obliki odprte kode za brezplačno uporabo z možnostjo nadaljnjega razvoja.

Ključne besede: simulacije, meritve, arhiviranje podatkov, HDF

Exchanging and storing simulation data in telecommunications with the software tool CostGlue

Extended abstract. Though storing and exchanging simulation data is a rather simple task done by simulation practitioners, it is quite often a challenge as huge quantities of data are not uncommon, and conversion between different formats can be much time consuming. After examining some of the needs of the telecommunications simulation community, we describe the architecture of a working prototype [3] of a general-purpose archiver and converter for large quantities of simulation data to be released as free software.

We describe the motivation, design issues and approach to the implementation of a low-overhead software package that can import simulation or measurement results into a common data structure, launch post-processing applications on the imported data, and store data or export it into various output formats.

Modern simulation packages and statistical tools use different, even proprietary formats, for storing results. This can be a major obstacle for an efficient exchange of scientific data. Moreover, if we consider issues like storage and data management, documentation and meta description, analysis and display or data filtering, the need for a common format or a tool to handle simulation results becomes apparent. This topic has been addressed in the framework of the European COST 285 [2] Action "Modeling and Simulation Tools for Research in Emerging Multi-service Telecommunications", a forum where researchers from all around Europe periodically meet to address issues related to the simulation of communications systems. The point made was that apparently no general purpose tools exist for exchanging big quantities of simulation data coming from different sources in different formats. Not only was the need for a common format for exchanging data highlighted, but also the need of feeding this data to different tools for post-processing, each requiring a different input format.

Keywords: simulation, measurements, data, archiving, HDF

Prejet 30. januar, 2007
Odobren 27. februar, 2007

1 Uvod

Moderni simulacijski paketi in statistična orodja uporabljajo različne, pogosto svoje lastne formate za shranjevanje simulacijskih podatkov. To lahko pomeni precejšnjo oviro za učinkovito izmenjavo znanstvenih podatkov. Če se ob tem zavedamo tudi drugih potreb, kot so: potreba po shranjevanju podatkov in upravljanju le-teh, dokumentiranju in uporabi metapodatkov ter analizi, filtriranju in prikazu podatkov, postane očitna tudi potreba po skupnem podatkovnem formatu in orodju za delo s simulacijskimi podatki [1]. Na področju telekomunikacij vključuje opisano tematiko evropski znanstveni projekt COST 285 – "Modeling and Simulation Tools for Research in Emerging Multi-service Telecommunications" [2], raziskovalni program "Algoritmi in optimizacijski postopki v telekomunikacijah" ter program CNR/MIUR "Legge 449/97 (projekt IS-Manet)". Projekt COST 285 je obenem tudi forum, v okviru katerega se raziskovalci različnih evropskih držav periodično sestajajo in obravnavajo probleme, povezane z metodologijo simulacij komunikacijskih sistemov. Forum je prišel do sklepa, da ni na voljo nobenega splošnega orodja za izmenjavo velikih količin simulacijskih podatkov različnega izvora, ko so zapisani v različnih podatkovnih formatih. Poleg potrebe po uvozu podatkov v skupni podatkovni format je forum poudaril tudi potrebo po izvozu teh podatkov v različne formate in s tem podporo obstoječim orodjem za njihovo obdelavo.

Programsko orodje CostGlue [3] je centralni repozitorij za podatke, ki so bili ustvarjeni z različnimi simulacijskimi orodji, in pretvornik različnih izhodnih formatov v različne vhodne formate.

Naslednja poglavja podrobneje opisujejo arhitekturo aplikacije CostGlue, še posebej njenega jedra – CoreGlue.

2 Model simulacijskega procesa

Za splošen in sistematični pregled nad ustvarjanjem, pretokom in obdelavo podatkov smo definirali referenčni model simulacijskega procesa, prikazanega na sliki 1. Model predstavlja razslojitev najpomembnejših funkcij, ki jih ponavadi potrebujemo pri simulacijah. Model tvorijo trije sloji in en opcijski podsloj.

Prvi sloj – *izvorni sloj* – so surovi simulacijski podatki, ki podrobno opisujejo posamezen zagon simulacije. Surovi simulacijski podatki so generirani z diskretno dogodkovnimi simulatorji (npr. ns-2, Opnet). Posamezen simulacijski dogodek ustvari enega ali več zapisov v izvornem sloju. Struktura in format sta odvisna od uporabljenega simulacijskega paketa. Najpogosteje sta v obliki velikih tabelaričnih sledilnih zapisov, v ASCII ali binarnem formatu. Opcijski prekodirni podsloj (1a. na sliki 1) upravlja surove podatke. Njegove glavne značilnosti so: pretvorba med različnimi formati (npr. ASCII v binarni format ali nasprotno), stiskanje podatkov (npr. gzip, Bzip2) in odstranjevanje zasebne informacije (npr. vrstice zaglavja iz sledilnega zapisa). Izvorni sloj podpira tako surove simulacijske zapise kot tudi zapise realnih meritev. Enak model namreč lahko uporabimo tudi pri analizi prometnih sledi realnih omrežij. V tem primeru surovi podatki niso rezultat simulacij, temveč podatkovne sledi, zajete na omrežni povezavi ali napravi. Razen razlike v orodjih za zbiranje podatkov (npr. Tcpdump ali Ethereal), ki jih tu uporabimo namesto simulatorja, ostajajo funkcije posameznih slojev enake.

Slika 1. Referenčni model simulacijskega procesa
Figure 1. Reference model of a simulation process

Drugi sloj – *obdelovalni sloj* – je zadolžen za analizo simulacijskih podatkov. Na tem sloju lahko pridemo do kumulativnih rezultatov, ki izhajajo iz surovih podatkov (npr. izračun povprečne zakasnitve paketov) ali statističnega zaupanja. Pomembna značilnost na obdelovalnem sloju je, da je količina podatkov, ki jih dobimo kot rezultat naknadne obdelave, bistveno manjša od količine podatkov na izvornem sloju.

Tretji sloj – *predstavitveni sloj* – je najvišji sloj. V njem so podatki organizirani v obliki, primerni za sporočanje najpomembnejših ugotovitev. Kadar zadoščajo preprosti tabelarični izpisi rezultatov, ta sloj ni uporabljen oziroma opravlja zgolj trivialne spremembe (npr. pretvorba formata števil, sprememba razmika med stolpci ipd.). Dokaj pogosto pa so podatki prikazani v dvo- ali tridimenzionalnih grafih ali celo v animirani obliki (npr. ns-2 NAM – Network AniMator). Za ta sloj sta pomembni predvsem dve značilnosti: fleksibilnost predstavitev in zmožnost ustvarjanja novih ali prilagajanja obstoječih predstavitvenih objektov.

Funkcionalnost posameznega orodja, uporabljenega pri simulaciji, lahko preslikamo v sloje opisanega modela. Orodje lahko zajema enega ali več slojev. V najboljšem primeru bi orodje zajemalo vse potrebne funkcije. V praksi je to zelo redko. Ponavadi uporabljamo množico komplementarnih orodij, ki skupaj obdelajo zahtevani obseg funkcij. Izbira orodij je dokaj poljubna. Odvisna je od zmogljivosti posameznega orodja, njegove razpoložljivosti in tudi od izkušenj, ki jih ima raziskovalec z določenim orodjem.

3 Formati za izmenjavo podatkov

Pri razvoju CostGlue je bilo zelo pomembno poznavanje različnih formatov, ki so v splošni rabi na področju telekomunikacij. Pri tem smo se oprli predvsem na informacije, ki smo jih pridobili od članov skupine COST 285, kar vključuje predstavnike več kot desetih evropskih držav. Podatki kažejo, da nobeno simulacijsko orodje nima dominantnega položaja in da obstaja velika raznolikost pri uporabi orodij.

Oglejmo si nekaj zanimivih rezultatov, ki smo jih dobili na podlagi ankete, izvedene med člani skupine COST 285:

- Najpogostejši način organizacije podatkov je tabelaričen. Drugi načini organizacije podatkov, kot na primer hierarhična, se redko uporabljajo.
- Podatki so večinoma uporabljeni v statistične namene ali za izris grafov. Drugi načini uporabe, kot na primer za rudarjenje podatkov, so zelo redki.
- Najpogostejši metodi za doseganje statistične natančnosti sta uporaba neodvisnih ponovitev in ekvivalentna korelacijska dolžina (posamezen zagon simulacije).
- Posamezen zagon simulacije ustvari podatke v velikosti od 1 MB do 2 GB. Individualna skupina simulacij pomeni od 1 do 100 zagonov simulacij. Pri meritvah štejemo kot skupino meritev od 1 do 5 meritev, kjer posamezna meritev pridela od 100 MB do 50 GB podatkov.
- Zasedenost prostora na trdem disku se giblje med 1 GB do 10 GB, pri dolgotrajnem shranjevanju pa je lahko tudi več.
- Uporabljeni metapodatki vključujejo tip, datum, vrednosti in opis parametrov, sledenje verzij, opis konfiguracije, simulacijske skripte in lokacijo.
- Metapodatki so lahko shranjeni na različnih lokacijah. Shranjeni so lahko v posebnih imenikih, v datotekah skupaj s podatki ali v ločenih datotekah, na ločenih pomnilniških lokacijah ali v ločenih skladovnicah podatkov.

Ugotovitve, ki imajo sicer omejen obseg, kažejo, da se najpogosteje uporablja tabelarični ASCII zapis. Sposobnost branja in zapisa v tabelarični ASCII podatkovni format je torej osnovna zahteva za predlagani sistem arhiviranja in pretvorbe podatkov. Kljub temu pa je potreben tudi splošnejši način branja in zapisa različnih formatov, kar omogoča modularna zgradba aplikacije CostGlue.

Kot smo že omenili, imajo simulacijski podatki in meritve veliko skupnega. Orodje, ki je uporabno za simulacije, je tako uporabno tudi pri meritvah, vendar so te ponavadi v različnih formatih, saj so za izvajanje simulacij uporabljena popolnoma druga orodja kot za izvajanje meritev. Vhodni pretvornik omogoča pretvorbo podatkov, ki prihajajo bodisi iz simulacij bodisi od meritev v enoten format in tako v njihovo poenoteno obravnavo. Zato že prvi prototip aplikacije CostGlue omogoča pretvorbo podatkov formatov ns-2 [4] in Tcpdump [8] v skupni format in nasprotno.

4 Skladovnica podatkov

Enoten način shranjevanja podatkov rešuje številne probleme pri izmenjavi simulacijskih podatkov. Naredili smo temeljito analizo obstoječih rešitev. Upoštevali smo uporabo preprostih tekstovnih formatov, XML, skladovnic podatkov, kot je SQL, ter specializiranih formatov s pripadajočimi knjižnicami. Ugotovili smo, da so najustreznejši specializirani formati. Pri tem smo se osredotočili predvsem na naslednje formate: HDF4, HDF5, netCDF, ODB, FITS in OpenDX. Za uporabo smo izbrali datotečni format HDF5 (Hierarchical Data Format) [5], ki se je izkazal za najprimernejšega, saj podpira vse zahteve za ustrezno organizacijo podatkov, pripadajoče knjižnice pa so na voljo v odprti kodi.

4.1 Struktura skladovnice podatkov

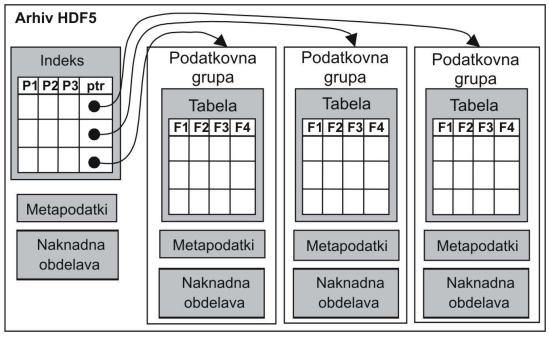
Večina simulacijskih podatkov pri telekomunikacijah so zbirke tabel z numeričnimi podatki. Posamezna tabela se nanaša na določene parametre, ki so specifični prav za simulacijski zagon, ki jo je ustvaril, ter je tako unikatno identificirana z vrednostmi danih parametrov.

Programska knjižnica, ki jo vključuje format HDF5, je sestavljena iz dveh osnovnih objektov: *podatkovne zbirke* in *podatkovne grupe*. Podatkovna zbirka je večdimenzionalno polje podatkovnih elementov, ki lahko vsebujejo različne podatkovne tipe. Podatki v podatkovnih zbirkah so lahko homogeni (samo en podatkovni tip v eni podatkovni zbirki – *preprosta podatkovna zbirka*) ali nehomogeni (različno število podatkovnih tipov v eni podatkovni zbirki – *sestavljena podatkovna zbirka*). Ker podatki, zbrani iz različnih simulatorjev, ponavadi vsebujejo različne tipe (npr. cela števila, realna števila, znaki ipd.), uporabljamo sestavljene podatkovne zbirke. Grupa HDF5 je struktura, ki je lahko prazna ali pa vsebuje enega ali več objektov HDF5. Z uporabo obeh osnovnih objektov HDF5 lahko podatke organiziramo hierarhično v obliki drevesne strukture. Iz glavne korenske grupe "\" lahko tako izpeljemo poljubno število objektov HDF5. Podatkovne grupe in podatkovne zbirke si lahko predstavljamo kot imenike in datoteke hierarhičnih datotečnih sistemov.

Da bi zadostili zahtevam učinkovitega shranjevanja podatkov, še posebej pri upravljanju podatkov, razumevanju in ponovni uporabi znanstvenih podatkov, ima lahko vsak objekt HDF5 pripadajoče metapodatke, shranjene v obliki atributov. Atributi so ponavadi majhne podatkovne zbirke, vezane na določeno grupo ali podatkovno zbirko. Atributi opisujejo naravo in/ali predvideno uporabo objekta, na katerega so vezani.

Pri načrtovanju strukture skladovnice podatkov smo želeli doseči fleksibilno predstavitev shranjenih

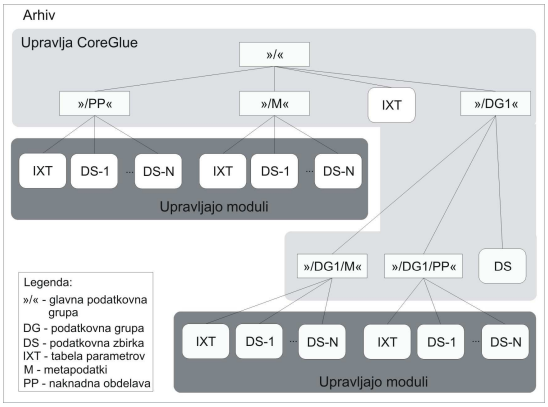
simulacijskih podatkov z uporabo enega večdimenzionalnega polja, v katerem lahko uporabnik izbere zgolj želeni del podatkov. Kljub temu, da HDF5 podpira večdimenzionalna polja, se shranjevanje velike količine podatkov v eno samo polje ni izkazalo kot učinkovito. Ker so podatkovni objekti v formatu HDF5 hierarhično organizirani, bi s shranjevanjem v enem večdimenzionalnem polju izgubili fleksibilnost, ki jo to ponuja. Zato smo vpeljali tabelo parametrov, ki preslika logični pogled večdimenzionalne matrike v hierarhično strukturo HDF5, kot to prikazuje slika 2.



Slika 2. Logična struktura skladovnice podatkov
Figure 2. Logical structure of the database

Slika 3 prikazuje podrobni vpogled v predlagano strukturo skladnice podatkov. Indeksiranje s pomočjo tabele parametrov je logični del, vse druge grupe in podatkovne zbirke pa dejanske podatke. Med dejanske podatke spadajo surovi simulacijski podatki, metapodatki in podatki naknadne obdelave. Celotna skladnica podatkov se obnaša enako kot posamezen arhiv. Vse grupe in podatkovne zbirke tvorijo dvonivojsko drevo, ki izhaja iz korenske grupe.

Neposredna razširitev shranjevanja podatkov v tabelah samo z numeričnimi podatki je uporaba tabel fiksne dolžine z različnimi tipi vrednosti (zastavice, številke, nizi). Knjižnica PyTables [6] podpira format HDF5 ter se odlikuje z dobrimi lastnostmi pri manipuliranju z dvodimenzionalnimi sestavljenimi podatkovnimi zbirkami HDF5 (v nadaljevanju tabele). V [6] najdemo tudi opravljene teste zmogljivosti. Vsaka tabela skupaj z metapodatki in podatki naknadne obdelave je povezana v grupo s podatki enega zagona simulacije. Podatkovne grupe so indeksirane z vektorji parametrov (posamezni stolpci). Celotni indeks je dvodimenzionalna matrika, ki ji pravimo *tabela parametrov*, v kateri se nahajajo parametri relevantni za posamezno shranjeno podatkovno grupo. Posamezen stolpec je vrsta parametra, vsaka vrstica pa vsebuje vrednosti parametrov, ki enolično zaznamujejo grupo. Tabela, ki je povezana v grupo, vsebuje podatke simulacije, organizirane v vrstice z različnimi polji (stolpci F1, F2, F3 in F4 na sliki 2).



Slika 3. Struktura skladovnice podatkov HDF5
Figure 3. HDF5 database structure

Celotna struktura je tako zbirka dvodimenzionalnih tabel, ki so indeksirane s polji P parametrov. To si lahko logično predstavljamo kot matriko z dimenzijami P + 2. Prvih P dimenzij je redkih, zadnji dve pa gosti. Z začetnimi P indeksi identificiramo podatkovno grupo, s preostalima dvema pa stolpec in vrstico v tabeli.

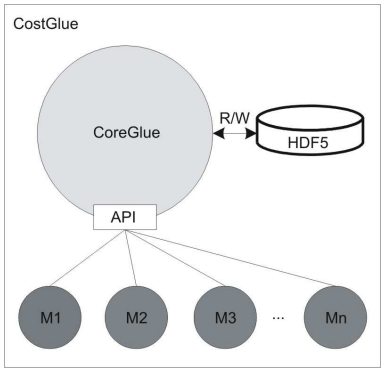
Kot realen primer si oglejmo simulacijo paketnih stikal v simulatorju ns-2. Vsak zagon simulatorja se razlikuje po vrednosti parametrov. Ti so lahko na primer tip arhitekture, velikost medpomnilnika, število vhodno/izhodnih vrat in prometna obremenitev. Kadar shranjujemo rezultate enega zagona simulacije v arhiv, se za to ustvari nova vrstica v tabeli parametrov. Vrednosti posamezne vrstice unikatno označujejo posamezen zagon simulacije. Vsaka vrstica dodatno vključuje celotno pot do podatkovne grupe, v kateri so rezultati zagona simulacije, metapodatki in podatki naknadne obdelave, kot to prikazuje slika 3. Pri metapodatkih govorimo o zapisu informacije o vrsti uporabljene skripte, tipu omrežne topologije, prometnih vzorcih ipd. Podatki naknadne obdelave vsebujejo informacijo o verjetnosti izgube paketa ter maksimalni, minimalni in povprečni zakasnitvi paketov. Metapodatki in podatki naknadne obdelave se nanašajo tudi na celoten arhiv, zato imajo za to predvideno mesto za shranjevanje.

5 Arhitektura aplikacije CostGlue

Jedrni del aplikacije CoreGlue poskrbi za indeks in strukturo skladovnice podatkov, vključno s tabelami. Posamezni moduli so zadolženi za metapodatke in vsebino naknadne obdelave, tako za posamezne podatkovne grupe, kot tudi za celoten arhiv. Jedro, skupaj z moduli, sestavlja celotno aplikacijo, ki smo jo poimenovali CostGlue. Arhitekturo aplikacije prikazuje slika 4. Iz nje je razvidno, da za branje in pisanje ter dinamično nalaganje modulov poskrbi jedrni del aplikacije. Sami moduli so namenjeni specializiranim nalogam. Moduli komunicirajo z jedrom prek

programskega vmesnika API (ang. Application Programing Interface). Primeri nalog, ki jih lahko opravljajo moduli:

- uvoz/izvoz podatkov v/iz arhiva,
- statistični izračuni nad podatki, shranjenimi v arhivu,
- izbiranje podatkov iz arhiva z uporabo kompleksnih filtrov,
- transformacije nad podatki v arhivu,
- grafični vmesnik za pregled podatkov v arhivu in
- specifični grafični vmesnik.

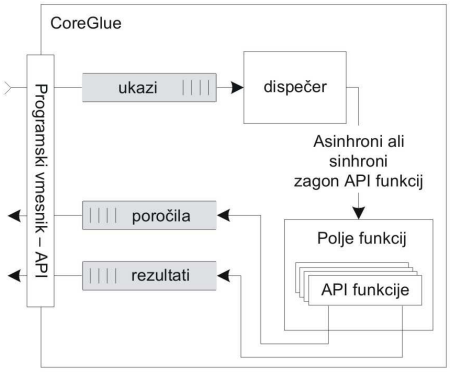


Slika 4. Arhitektura aplikacije CostGlue
Figure 4. CostGlue application architecture

6 Programski vmesnik modula

Moduli so samoopisljivi programi, napisani v programskem jeziku Python. Jedro CoreGlue lahko poišče razpoložljive module ter izvede poizvedbe nad njimi, da ugotovi njihove sposobnosti. V modulih so zapisani opisi parametrov, ki jih potrebujejo, ter vrsta izhoda, ki ga ustvarijo. Metode, ki so del programskega vmesnika, omogočajo manipulacijo z indeksi. Na tej podlagi lahko dodajamo ali odstranjujemo podatkovne grupe, manipuliramo s podatkovnimi grupami, to je: dodajamo ali odstranjujemo surove podatke, metapodatke in podatke naknadne obdelave.

Programski vmesnik vključuje tudi metode za dostop do podatkov s t.i. selektorji z indeksno notacijo. Tak način je zastopan pri matričnih izračunih, ki jih podpirata programa Matlab in Octave [7]. Pri tej notaciji pride v poštev vsak od P + 2 indeksov, pri čemer označuje "1", najmanjši indeks, "end" največji indeks, "n:m" vse indekse med vključno *n* in *m*, ":" indicira vse, "n:s:m" območje med *n* in *m* s korakom *s*, s poljem "[1 5 6 8]" pa označimo le izbrane indekse. Z uporabo indeksne notacije lahko operiramo z ortogonalnimi deli večdimenzionalne matrike, ki jo sestavljajo podatki celotne skladovnice podatkov.



Slika 5. Interakcija med programskim vmesnikom – API in jedrom CoreGlue
Figure 5. API and CoreGlue interaction

Slika 5 prikazuje jedro (CoreGlue), ki vsebuje tri vrste čakalnih vrst za izmenjavo različnih sporočil v obliki ukazov, rezultatov ali poročil. Dani modul lahko dela samostojno ali pa je odvisen od drugih. Na primer, modul za interaktivno ukazno vrstico lahko pokliče drug specializiran modul, ki opravi določeno delo. Moduli se lahko razlikujejo tudi po načinu sestave. Preprosti moduli zatem, ko pošljejo ukaz jedru, čakajo na rezultate. Tako v tem času ne morejo opravljati drugega dela. Alternativa temu so kompleksni moduli, ki delujejo asinhrono. Ti lahko periodično preverjajo čakalno vrsto z rezultati in informirajo uporabnika o delovnem napredku.

Ukazi programskega vmesnika se delijo na dve vrsti: sinhrone in asinhrone. Pri sinhronih ukazih, v nasprotju z asinhronimi, je onemogočena možnost prikaza poročil napredovanja in med izvajanjem ukaza ni mogoče odložiti, povrniti ali ustaviti njegovega delovanja. Ob klicu ukaza programskega vmesnika jedro ustvari sporočilo, ki se uvrsti v ukazno čakalno vrsto. Za branje sporočila in izvršitev ukaza skrbi dispečer, ki je delujoča zanka, zagnana z ločeno procesorsko nitjo. Slika 6 skicira implementacijo dela programskega vmesnika v jedru.

6.1 Modul za upravljanje s CoreGlue

Pri zagonu jedra CoreGlue iz ukazne vrstice je prvi argument ime modula. Nadaljnji argumenti so opcijski in pripadajo modulu. Če imamo na mestu prvega argumenta zapisano vrednost **html**, se CoreGlue obnaša kot strežnik HTTP, ki ponuja grafični vmesnik. Skozi ta vmesnik lahko uporabnik pregleduje seznam razpoložljivih modulov. Za vsak modul najdemo njegov splošni opis, potrebne vhodne podatke in vrsto izhoda, ki ga proizvede.

Trenutno je že izvedena interaktivna ukazna vrstica. Ta modul se uporablja za potrebe razhroščevanja jedra in drugih modulov. Uporabi se lahko tudi za uvoz in izvoz tabelaričnih podatkov. Specializiran modul

avtomatsko razpoznava tip sledenja in opravlja avtomatsko poimenovanje ter po potrebi tudi filtriranje.

7 Sklep

Programsko orodje CostGlue za pretvorbo in shranjevanje simulacijskih podatkov, ki je opisano v tem članku, olajša izmenjavo in omogoči učinkovitejše upravljanje podatkov med raziskovalci. Dodatno želimo olajšati delo pri uporabi različnih simulacijskih, merilnih, podatkovno obdelovalnih in predstavitvenih programskih orodij, ki imajo različne vhodne in izhodne podatkovne formate. Izdelava aplikacije je v teku. Trenutno je na voljo delujoč prototip in vmesnik za razhroščevanje. Takoj ko prototip prestane vse testne stopnje, ga bo mogoče − zahvaljujoč njegovi modularni zgradbi − obogatiti z dodatno funkcionalnostjo, pri čemer bodo lahko sodelovali drugi raziskovalci. Morebitne razširitve trenutne funkcionalnosti vključujejo dodajanje možnosti za delo z netabelaričnimi podatki in definicijo uporabne strukture za metapodatke in podatke naknadne obdelave.

Razvito programsko opremo kot del raziskovalne naloge bomo izdali z brezplačno licenco. Izbirali bomo med licencami MIT X, GNU LGPL in GNU GPL, za katere menimo, da so najprimernejše za izdajo brezplačne raziskovalne programske opreme.

8 Literatura

[1] Gray, J., Liu, D., DeWitt, D., Heber G. Scientific Data Managment in the Comming Decade, ACM SIGMOD Record, Let. 34.4, str. 35-41, 2005.

[2] Bragg, A. Observations and thoughts on the possible approaches for addressing the tasks specified in the COST 285 work-plan. COST 285 začasni dokument TD/285/03/15, CNUCE-CNR (IT), 2004.

[3] Domača stran programskega orodja CostGlue: http://wnet.isti.cnr.it/software/costglue

[4] McCanne, S., and Floyd, S. The network simulator - ns-2. University of Berkley, 2005.

[5] Folk, M., McGrath, R., and Yeager, N. HDF: an update and future directions. Mednarodni simpozij: Geoscience and Remote Sensing Symposium (IGARSS'99), IEEE izdaja, Let. 1, str. 273–275, 1999.

[6] PyTables programski paket: http://www.pytables.org/moin

[7] Golub, and Loan, V. Matrix Computations, tretja izdaja, The Johns Hopkins University Press, 1996.

[8] Domača stran programskega orodja Tcpdump: http://www.tcpdump.org/

Dragan Savić je leta 2004 diplomiral na Fakulteti za elektrotehniko (v Ljubljani). Kot mladi raziskovalec je na tej fakulteti zaposlen v Laboratoriju za komunikacijske naprave. Na raziskovalnem področju se ukvarja s komunikacijskimi omrežji, še posebej s simulacijami telekomunikacijskih sistemov.

Sašo Tomažič je zaposlen na Fakulteti za elektrotehniko v Ljubljani kot profesor in predstojnik Laboratorija za komunikacijske naprave. Njegovo sedanje delo zajema raziskave na področju obdelave signalov, varnosti v telekomunikacijah elektronskega poslovanja in porazdeljenih podatkovnih sistemov.

Janez Bešter je doktoriral leta 1995 in je zaposlen na Fakulteti za elektrotehniko v Ljubljani kot profesor in predstojnik Laboratorija za telekomunikacije. Njegovo raziskovalno, razvojno in pedagoško delo je povezano s področjem načrtovanja, realizacije in vodenja telekomunikacijskih sistemov in storitev ter uporabo informacijskih tehnologij in telekomunikacij na področju e-izobraževanja.

Matevž Pustišek je leta 1993 diplomiral, leta 1997 pa magistriral na Fakulteti za elektrotehniko Univerze v Ljubljani s področja telekomunikacij. Zaposlen je kot asistent v Laboratoriju za telekomunikacije na Fakulteti za elektrotehniko. V raziskovalnem delu se posveča študiju širokopasovnih telekomunikacijskih omrežij, sistemov in storitev. Poseben poudarek daje tudi analizi telekomunikacijskih omrežij in sistemov s pomočjo simulacij in razvoju internetnih rešitev s področja e-izobraževanja.