Ivan Skuliber, Tomislav Štefanec

Ericsson Nikola Tesla d.d., Zagreb, Hrvatska
Ericsson Nikola Tesla d.d., Zagreb, Croatia

PARALELNI SIP PARSER NA VIŠEJEZGRENIM PROCESORIMA

PARALLEL SIP PARSER ON MULTICORE PROCESSORS

Sažetak

Rad prikazuje paralelni parser za protokol uspostave sjednice (SIP). SIP je protokol koji koristimo u 3GPP-ovom radnom okviru višemedijskog podsustava zasnovanog na Internet protokolu (IP Multimedia System - IMS). Parser koristi višestruke jezgre višejezgrenog procesora i zasniva se na SIP-ovoj službenoj gramatici napisanoj u proširenom *Backus Naur* obliku, kojim se definiraju SIP-ove leksičke i sintaksne značajke. Rad detaljno opisuje algoritam parsera, neovisan o konkretnom programskom jeziku, te implementaciju algoritma u dva programska jezika namijenjena programiranju višejezgrenih procesora (C++ s CILK višejezgrenom bibliotekom i Java s JSR-166 višejezgrenom bibliotekom). Implementacije parsera ispitane su službenim SIP-ovim ispitnim porukama. Na kraju rada prikazana su eksperimentalna mjerenja te obrazložene prednosti i nedostaci pojedine implementacije.

Abstract

The paper presents a parallel parser for the Session Initiation Protocol (SIP), a protocol used in the 3GPP's IP Multimedia System (IMS) framework. The parser utilizes multiple cores of a multicore processor and is based on SIP's official Augmented Backus Naur Form (ABNF) grammar that specifies SIP's lexical and syntactic properties. The paper describes in detail the parser's algorithm, which is not bound to any programming language and its implementations in two multicore-oriented programming languages (C++ with CILK multicore library and Java with JSR-166 multicore library). The parser's implementations are tested with official SIP Torture suite of messages. Experimental results are shown and based upon them the strengths and weaknesses of each of the implementations are discussed.

KLJUČNE RIJEČI:	KEY WORDS:
Protokol za uspostavu sjednice (SIP) Paralelno parsiranje	Session Initiation Protocol (SIP) Parallel parsing
Višejezgreni procesori	Multicore processors

1 Uvod

Zahvaljujući sveprisutnosti i prihvaćenosti usluga zasnovanih na Internet protokolu (IP) sve veća je važnost protokola poput protokola za uspostavu sjednice (engl. Session Initiation Protocol, SIP). SIP je tekstualno zasnovan signalizacijski protokol za stvaranje, upravljanje, izmjenu i zatvaranje sjednica [1]. Osnovni je protokol u 3GPP-ovom IP Multimedia System (IMS) radnom okviru, a primarnu namjenu ima u uspostavi i upravljanju audio i video pozivima (npr. VoIP i telekonferencije), IPTV-u te instant porukama (engl. instant messaging).

Službena specifikacija SIP standarda sadrži formalnu gramatiku napisanu u proširenom Backus Naur obliku (engl. Augmented Backus Naur Form, ABNF) [2] čime je strogo definiran leksički i sintaksni zapis protokola. Takva je gramatika lako čitljiva i razumljiva, no istodobno zahtjeva iznimna računalna sredstava za validaciju ispravnosti SIP poruka. Postojeći radovi i istraživanja jasno pokazuju vremensku zahtjevnost i značajnu potrošnju računalnih sredstava prilikom obrade SIP poruka. Pri tome, trećina ukupnog vremena obrade SIP poruke troši se na postupak parsiranja SIP poruka [3, 4, 5]. Zbog toga je potencijalno ubrzanje postupka parsiranja SIP poruka itekako korisno, a korištenje višejezgrenih procesora je jedan od načina kojim se to ubrzanje može postići.

Parsiranje, u općenitom smislu, za dani tekstualni ulaz radi analizu ulaznog teksta. Analiza se izvodi prema pravilima definiranim u formalnoj specifikaciji, tzv. formalnoj gramatici. Uspješna analiza završava organiziranjem ulaznog teksta u hijerarhijske cjeline definirane formalnom specifikacijom. Neuspješna analiza označava neslaganje ulaznog teksta s formalnom specifikacijom. Primijenjeno na općeniti tekst, parsiranje bi tekst organiziralo u hijerarhijske cjeline od poglavlja, odjeljaka, paragrafa i rečenica sve do imenica, glagola, priloga, prijedloga i interpunkcijskih znakova. U pogledu SIP poruke, parsiranje analizira poruku u cilju prepoznavanja hijerarhijski organiziranih cjelina poruke pri čemu su te cjeline (SIP zahtijev, SIP odgovor, zaglavlja, oblik IPv4 i IPv6 adresa i dr.) formalno definirane SIP-ovom gramatikom. Osnovne informacije o parsiranju mogu se pronaći na stranicama [6].

U ovome radu opisan je dizajn i implementacija paralelnog SIP parsera namijenjenog izvođenju na višejezgrenim procesorima s ciljem poboljšanja radnih svojstava u usporedbi s tradicionalnim, jednojezgrenim SIP parserima. Potrebno je istaknuti kako su postupci parsiranja po prirodi slijedni postupci što otežava njihovu paralelizaciju. Zbog toga paralelizacija parsiranja ne jamči poboljšanje radnih svojstava [7]. Postojeći radovi s općeprihvaćenim SIP parserima potvrđuju ovu tvrdnju, jasno pokazujući da je razmjeran rast (engl. scalability) paraleliziranih SIP parsera već na višejezgrenim procesorima s tri i četiri jezgre manji od linearnog [8].

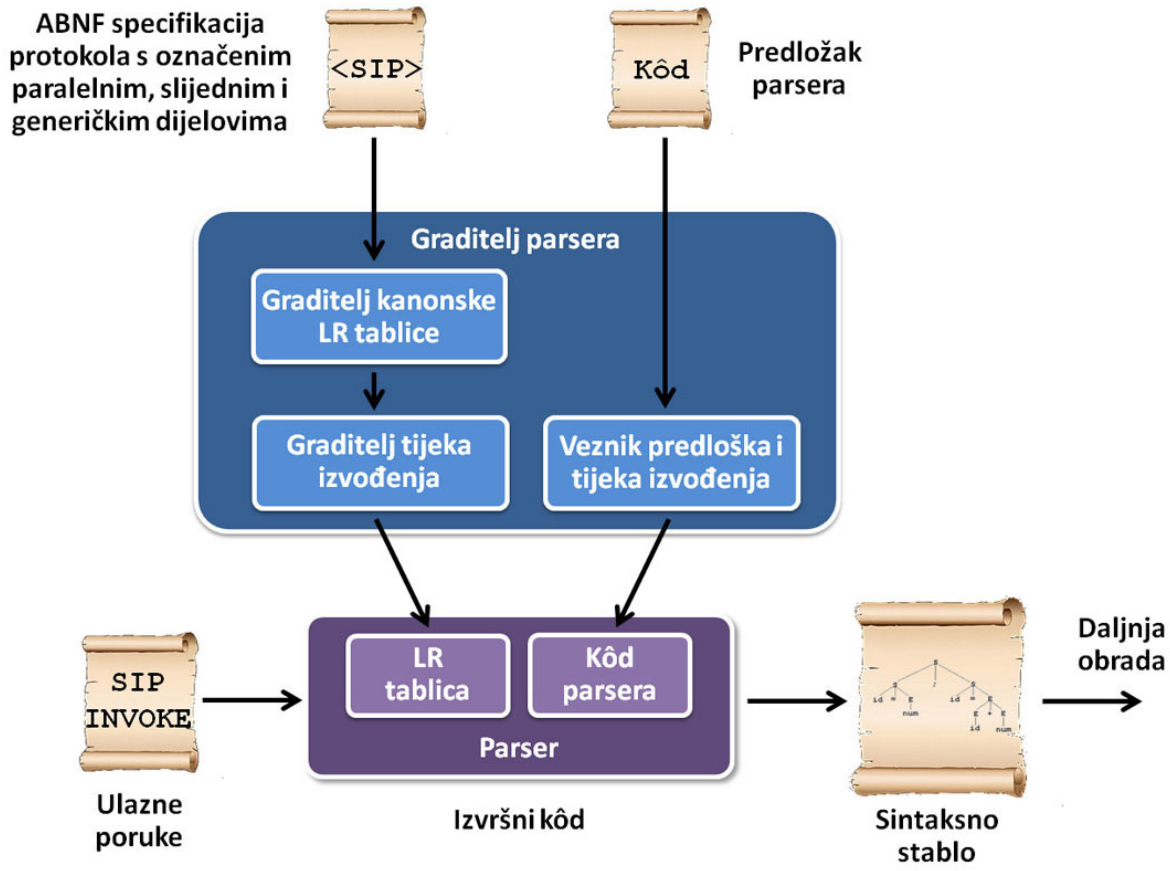
Implementacija paralelnog SIP parsera ostvarena je u programskom jeziku C++ pomoću CILK višejezgrene programske biblioteke [9] i programskom jeziku Java pomoću JSR-166 višejezgrene programske biblioteke [10]. Razlog odabira navedena dva programska jezika je njihova opća prihvaćenost te široka primjena, a razlog odabira navedenih višejezgrenih programskih biblioteka je mogućnost ostvarivanja paralelnih programa korištenjem jednostavnih programskih paradigmi i mehanizama.

2 Postupak izgradnje parsera

Za izradu paralelnog SIP parsera koristimo deterministički postupak generiranja kanonskog LR (eng. *Left to Right*) parsera na osnovi zadane gramatike parsera [7]. Navedeni postupak je izmijenjen u cilju stvaranja nedeterminističkih parsera na osnovi zadanih nedeterminističkih gramatika kao što su ABNF gramatike. Postupak generiranja kanonskog LR parsera je odabran jer prihvaća širok spektar potencijalnih formalnih gramatika i rezultira parserima s brzim vremenom izvođenja. Nadalje, navedeni postupak je jedan od osnovnih postupaka za prevođenje programskih jezika te su njegova svojstva dobro poznata.

Postupak izgradnje paralelnog SIP parsera prikazan je na slici 1. Ulazni parametri za izgradnju parsera su gramatička pravila u ABNF zapisu (u ovom slučaju, pravila SIP gramatike iz [1, 9, 10]) te predložak samog parsera.

Predložak parsera sadrži generički algoritam za parsiranje kojim se upravlja kroz podatke iz tablice parsera. Kao što je već naznačeno u uvodu, predlošci su ostvareni u programskom jeziku C++ uz korištenje CILK višejezgrene programske biblioteke te u programskom jeziku Java uz korištenje JSR-166 višejezgrene programske biblioteke. Predlošci su zapravo nedeterministički potisni automati koji čitaju znak po znak iz ulazne poruke te na osnovu pročitanog znaka i stanja internog stoga [11] prelaze u novo stanje. Prijelazom u novo stanje mijenja se stanje internog stoga te se parser pomiče na idući znak u ulaznoj poruci. Dodatno, predložak sadrži listu instrukcija u meta-jeziku koje definiraju pozive funkcija za određena gramatička pravila.



Slika 1: Postupak izgradnje paralelnog SIP parsera

Instrukcije u meta-jeziku se obrađuju u generatoru parsera, gdje se povezuju s odgovarajućim dijelovima LR tablice. U konačnici, na osnovu predloška parsera i LR tablice, generator parsera stvara izvorni kod parsera koji je vezan na tu specifičnu LR tablicu.

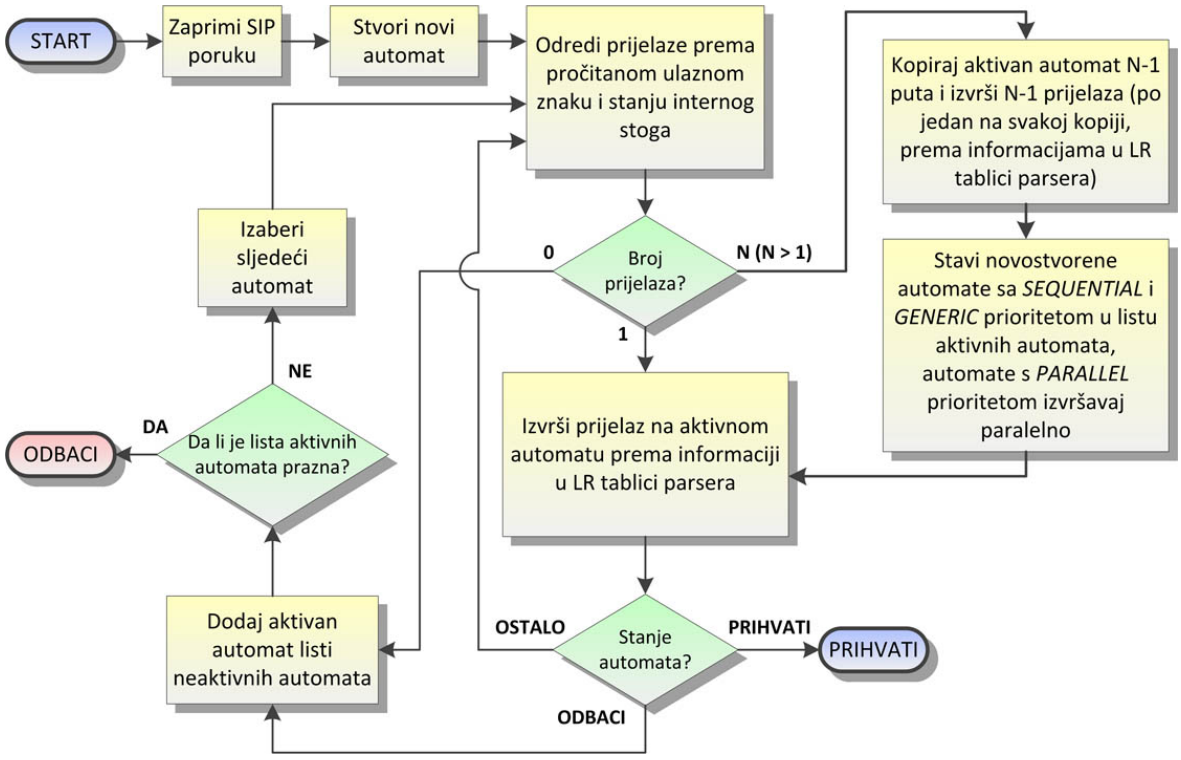
ABNF zapis gramatičkih pravila omogućava stvaranje kanonske LR tablice prema postupku opisanom u literaturi [7] uz dodatak proširenja koja omogućavaju nedeterminizam [12]. Dodatno, gramatička pravila proširena su s paralelizacijskim prioritetima, čime su naznačeni nedeterministički dijelovi gramatike koji se mogu paralelno obrađivati. Za SIP gramatiku razlikujemo tri tipa paralelizacijskih prioriteta. *PARALLEL* paralelizacijski prioritet označava gramatičko pravilo koje se može paralelno obrađivati. Na primjer, sva pravila vezana uz SIP zaglavlja označavamo *PARALLEL* paralelizacijskim prioritetom te na taj način omogućavamo da se svako pojedino zaglavlje može obrađivati potpuno neovisno o drugim zaglavljima i u isto vrijeme kad i druga zaglavlja. *SEQUENTIAL* paralelizacijski prioritet označava pravila koja se moraju obrađivati slijedno te da se njihova obrada ne može paralelizirati. Na primjer, sva pravila vezana uz leksičke jedinke (engl. token) moraju biti označena ovim paralelizacijskim prioritetom jer ona definiraju razmake, riječi i brojčane konstante koje koristimo svugdje u SIP gramatici. Kad ta pravila ne bi bila označena *SEQUENTIAL* paralelizacijskim prioritetom, tad bi se u vrlo kratkom vremenu stvorio velik broj dretvi (paralelnih procesa s gledišta višejezgrenih programskih biblioteka, engl. thread) koje bi zagušile današnje višejezgrene procesore. Treći paralelizacijski prioritet korišten u SIP gramatici je *GENERIC*. Ovaj prioritet je specifičan za SIP gramatiku i koristimoa ga za označavanje gramatičkih pravila koja služe proširivanju SIP-a i omogućavaju prihvaćanje gotovo bilo kakve ulazne poruke. Uobičajene SIP poruke ne koriste ova pravila, već se ona primjenjuju jedino kod manjeg broja specifičnih testnih poruka [13]. Zbog toga je *GENERIC* najniži paralelizacijski prioritet i označava pravila koja će se koristiti tek kad su sve druge mogućnosti iskorištene. Ovakav pristup može negativno utjecati na radna svojstva u slučaju parsiranja specifičnih testnih poruka, ali osigurava ispravno parsiranje svih SIP poruka napisanih u skladu sa SIP standardom.

Generirana LR tablica za SIP gramatiku je iznimno velika (više od 300 MB). Zbog toga ju je nužno optimirati korištenjem agregacije prijelaza pomoću grupa znakova, pri čemu se agregiraju prijelazi koji za različite ulazne znakove iz trenutnog stanja prelaze u isto stanje, te putem eliminacije višestrukih identičnih stanja i mogućih grana parsiranja. Nakon optimizacije, generirana LR tablica zauzima nešto više od 30 MB (deseterostruko smanjenje). Konačno, generator parsera računa *hash* vrijednost (mjesto u nizu elemenata

na kojem treba tražiti odgovarajuću vrijednost) za svako pravilo gramatike te organizira izračunate *hash* vrijednosti u četiri razine instrukcija tijeka izvođenja (engl. *control-flow statements*), čime se omogućava brz i učinkovit pristup svakom gramatičkom pravilu.

3 Algoritam paralelnog parsiranja

Paralelni SIP parser zasnovan je na generičkom algoritmu za parsiranje prikazanom na slici 2.



Slika 2: Algoritam paralelnog SIP parsera

Algoritam svoj rad započinje primanjem SIP poruke i stvaranjem novog nedeterminističkog potisnog automata. Zatim počinje sam postupak parsiranja, odnosno glavna petlja parsera. Postupak parsiranja čita znak po znak iz SIP poruke s ciljem završavanja čitanja cijele SIP poruke u trenutku kada se potisni automat nalazi u stanju kojim se označava ispravnost SIP poruke (npr. automat obradu završava u stanju prihvaćanja poruke). Prijelazi između stanja automata odvijaju se na osnovu pročitanog znaka ulazne poruke i vrijednosti internog stoga. Kod nedeterminističke gramatike, kao što je SIP gramatika, može se dogoditi da za pročitani znak i vrijednosti internog stoga postoji više mogućih prijelaza. Kod slijednih parsera moguće je izvršiti samo jedan prijelaz [12], dok se kod paralelnih parsera teoretski mogu izvršiti svi prijelazi stvaranjem odgovarajućeg broja dretvi od kojih svaka predstavlja jedan mogući prijelaz. Trenutačno je ova mogućnost paralelnih parsera još isključivo teoretska budući da stvaranje velikog broja dretvi može izazvati ozbiljno narušavanje radnih svojstava, zbog zagušenja procesora i memorije. Prema tome, potrebno je odabrati podskup prijelaza koji će se izvršiti paralelno, dok će se ostali prijelazi izvršiti slijedno. Prijelazi koji će se izvršiti paralelno i oni koji će se izvršiti slijedno odabiru se na temelju paralelizacijskih prioriteta opisanih u poglavlju "Postupak izgradnje parsera". Prijelazi označeni s *PARALLEL* izvršavaju se paralelno; svaka dretva dobiva kopiju potisnog automata, što uključuje stanje automata, stanje internog stoga i poziciju zadnjeg pročitanog znaka iz ulazne poruke te nad svojim automatom izvršava prijelaz. Prijelazi označeni sa *SEQUENTIAL* i *GENERIC* spremaju se u listu aktivnih automata i obrađuju se tek kad trenutačni automat odbaci poruku, odnosno kad iscrpi sve moguće prijelaze.

U cilju dodatnog smanjenja nedeterminizma parsiranja SIP poruka, koristi se lista neaktivnih automata. Lista neaktivnih automata sadrži *hash* vrijednosti automata koji su odbacili poruku. *Hash* vrijednosti automata se računaju u trenutku kad automat može izvršiti više od jednog prijelaza te se računaju na osnovu trenutačnene pozicije u ulaznoj poruci, zadnjeg pročitanog znaka i gornjih pet vrijednost internog stoga. Takva *hash* vrijednost omogućava provjeru prijelaza, čime izbjegavamo ponavljanje prijelaza koji su već prije bili ispitani. Lista aktivnih automata sadrži *hash* vrijednosti neobrađenih prijelaza (i odgovarajućih automata), čime izbjegavamo uvišestručavanje aktivnih automata.

Obrada SIP poruke završava ili prihvaćanjem ili odbacivanjem ulazne poruke. Poruka se prihvaća ako bar jedan potisni automat završi u stanju prihvaćanja, a odbacuje se ako svi automati završe u stanju odbacivanja i lista aktivnih automata je prazna.

3.1 C++ s CILK višejezgrenom bibliotekom

Implementacija paralelnog SIP parsera u programskom jeziku C++ sa CILK višejezgrenom bibliotekom izravno slijedi postupak parsiranja opisan u poglavlju "Algoritam paralelnog parsiranja". Glavna petlja parsera koja čita znak po znak iz ulazne poruke ostvarena je kao rekurzivna metoda koja se poziva kad treba pročitati sljedeći znak. Paralelizacija je ostvarena pomoću CILK-ove paralelne *for*-petlje (ključna riječ cilk_for) koja se koristi za paralelno izvršavanje svih prijelaza označenih *PARALLEL* paralelizacijskim prioritetom. Na taj način se rekurzivni pozivi svakog od ovih prijelaza izvršavaju paralelno. Prijelazi označeni sa *SEQUENTIAL* i *GENERIC* obrađuju se korištenjem normalne *for*-petlje programskog jezika C++ i svi rekurzivni pozivi se izvršavaju u istoj dretvi.

3.2 Java s JSR-166 višejezgrenom bibliotekom

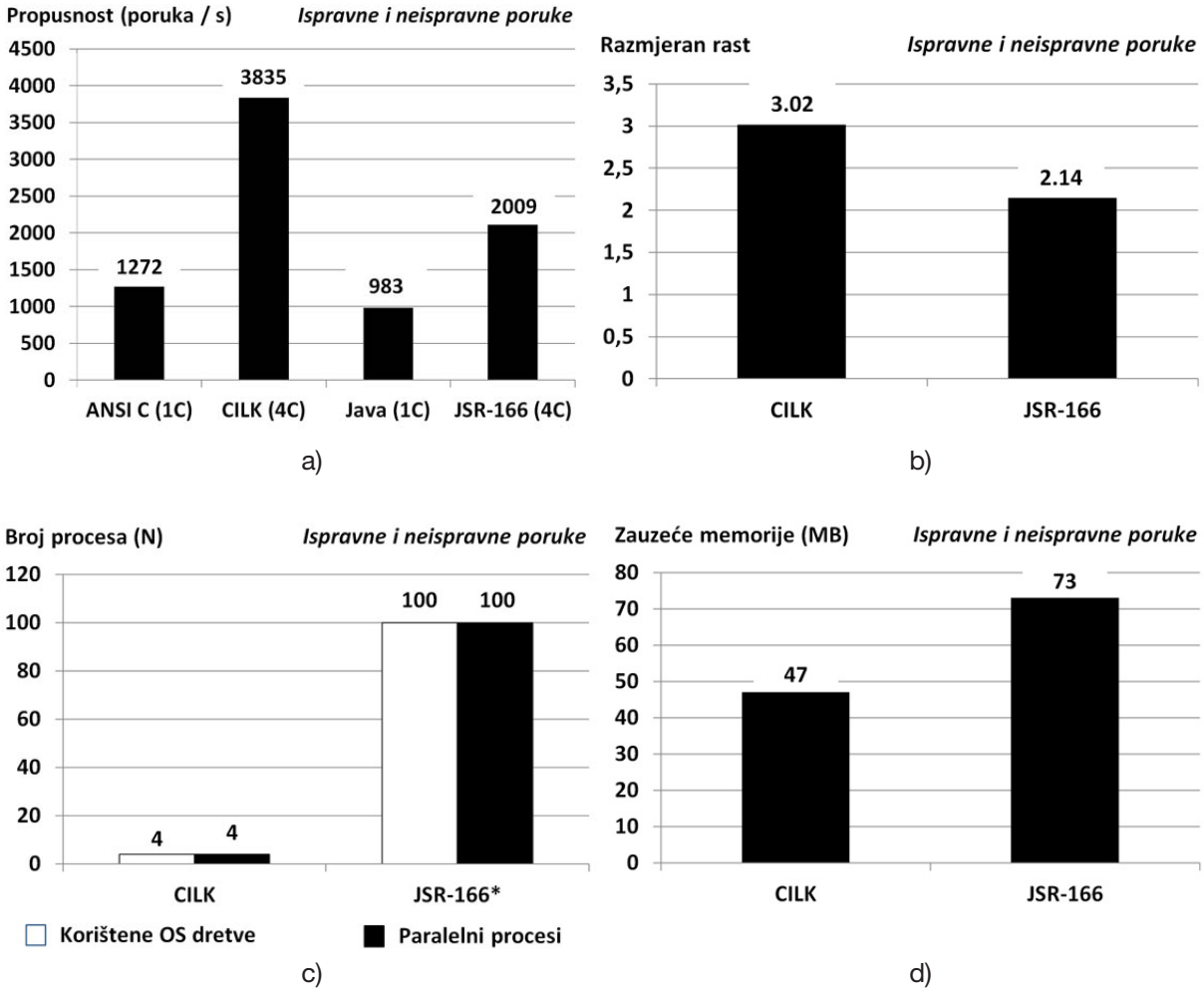
Inicijalna JSR-166 implementacija paralelnog SIP parsera koristila je ista načela kao i CILK implementacija. No, kako JSR-166 višejezgrena biblioteka ne ograničava maksimalni broj korištenih dretvi, bilo je potrebno uvesti mehanizam nadziranja broja trenutačno korištenih dretvi i ograničavanja maksimalnog broja dretvi u sustavu.

Glavna petlja parsera ostvarena je, kao i u slučaju CILK implementacije, kao rekurzivna metoda koja se poziva kod čitanja sljedećeg znaka ulazne poruke. Razlika postoji zbog nadzornog mehanizma koji za prijelaze označene s *PARALLEL* ne stvara nove dretve odmah, već ih dodaje u listu zadataka (tzv. *callable tasks* u Javi). Prije izvršavanja zadataka provjerava se trenutačni broj dretvi te se zadaci izvršavaju ako je taj broj dretvi manji od maksimalnog. U suprotnom, odgovarajući automati se dodaju u listu aktivnih automata i obrađuju se slijedno.

4 Eksperimentalna mjerenja

Za procjenu uspješnosti paralelizacije slijednog parsera izvršena je analiza radnih svojstava implementacije paralelnog SIP parsera. Mjerenja su provedena na računalu s četverojezgrenim Intel Q9400 procesorom frekvencije 2.66 GHz i 4 GB radne memorije na kojem je instaliran Windows 7 SP 1 operacijski sustav. Procesor nema *hyper-threading* mogućnosti te je maksimalan broj simultanih dretvi 4.
Paralelni SIP parser ostvaren uz pomoć CILK-a koristi zadnju verziju CILK-a tvrtke Cilk Arts Inc. To je verzija prije nego je CILK integriran u Intelove proizvode za razvoj paralelnog softvera, dok paralelni SIP parser ostvaren u Javi koristi JSR-166 verzije "y". Mjerenja su provedena korištenjem 500 različitih poruka, što ispravnih, što neispravnih, prikupljenih iz uobičajenih poruka iz literature [1] te korištenjem testnih poruka iz literature [13, 14] i posebno osmišljenih varijanti tih poruka. Eksperiment je izvršavan tijekom 30 sekundi unutar kojih se ciklički obrađuje navedenih 500 poruka. Poseban naglasak u implementaciji paralelnih SIP parsera stavljen je na potpuno poštivanje formalne SIP gramatike [1], čime je osigurano točno prepoznavanje ispravnih i neispravnih SIP poruka što uključuje i specifične testne poruke.

Slika 3.a prikazuje propusnost svih implementacija paralelnih SIP parsera. Na slici su vidljiva i radna svojstva slijednih parsera implementiranih u ANSI C-u i Javi. Slijedni parseri koriste isti dizajn kao i paralelni parseri, osim što se se nikad ne stvaraju nove dretve već se koristi lista aktivnih automata u svim slučajevima kada bi se pokrenule nove dretve u paralelnim parserima [12].

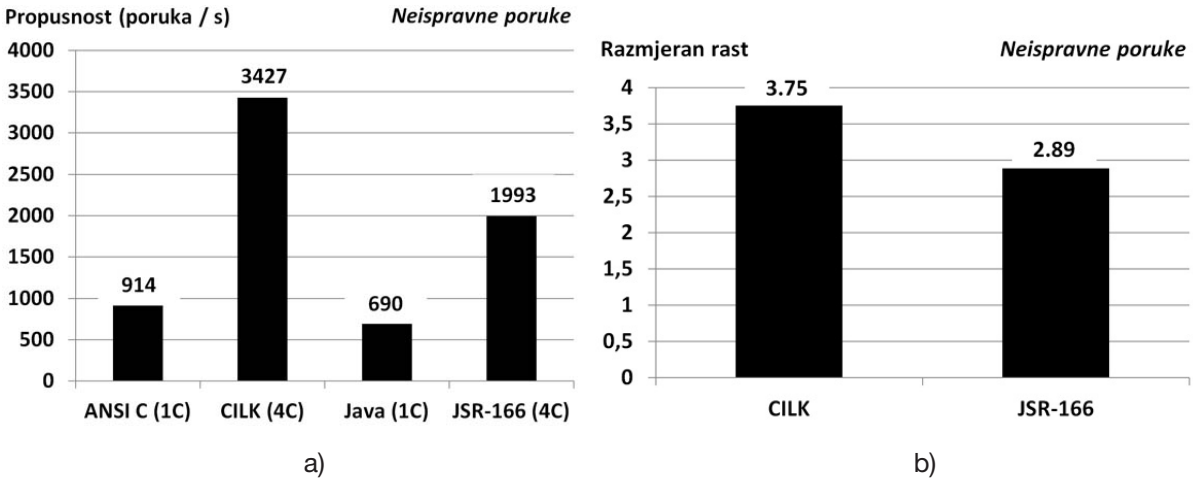


Slika 3: Analiza radnih svojstava implementacije paralelnog SIP parsera: a) propusnost; b) razmjeran rast paralelnog SIP parsera prilikom izvođenja na četverojezgrenom procesoru u odnosu na slijedni parser; c) usporedba korištenih dretvi operacijskog sustava u odnosu na korištene paralelne procese u višejezgrenoj biblioteci (* JSR-166 implementacija je programski ograničena na korištenje 100 paralelnih procesa); d) prosječno zauzeće radne memorije za vrijeme eksperimenata

CILK implementacija je dvostruko brža od JSR-166 implementacije. To je očekivani rezultat budući da JSR-166 implementacija koristi Java virtualni stroj te se izvršava kao interpretirani kôd, dok je CILK implementacija prevedena u strojni kôd, specifičan za korišteni procesor. U svojstvu razmjernog rasta CILK implementacija pokazuje rast manji od linearnog, ali istovjetan razmjernom rastu *open-source* SIP parsera koji su danas u upotrebi [8]. JSR-166 implementacija pokazuje još slabiju skalabilnost te samo dvostruki rast u odnosu na slijedni parser.

Kako bismo u potpunosti razumjeli rast manji od linearnog moramo analizirati radne karakteristike paralelnih SIP parsera. Potrebno je odrediti broj dretvi koje se stvaraju za vrijeme parsiranja te način njihovog raspoređivanja na dretve operacijskog sustava (slika 3.c). CILK implementacija u prosjeku koristi broj dretvi koji odgovara broju jezgara u procesoru, što znači da je svaki paralelni proces raspoređen na dretvu operacijskog sustava. JSR-166 implementacija na jednak način raspoređuje paralelne procese na dretve operacijskog sustava, ali ne ograničava ukupan broj dretvi u sustavu, što može dovesti do zagušenja procesora. Čak i za jednostavne SIP poruke, dužine svega 6-7 linija, JSR-166 implementacija stvara preko 400 paralelnih procesa koji zauzimaju jednak broj dretvi operacijskog sustava, što uzrokuje prestanak rada Java virtualnog stroja. Zbog toga je u JSR-166 implementaciji maksimalan broj dretvi ograničen na 100, što se pokazao kao izbor koji omogućava najbolju propusnost ove implementacije. Nadalje, lošiji razmjeran rast JSR-166 implementacije može se pripisati razlici u upravljanju dretvama između CILK i JSR-166 biblioteka.

Memorijski zahtjevi CILK implementacije su u prosjeku 50 MB radne memorije, a JSR-166 implementacije su prosječno 70 MB (slika 3.d), što je dvostruko više od slijednog ANSI C SIP parsera koji koristi svega 30 MB memorije, što je otprilike jednako veličini optimirane LR tablice za SIP [12].



Slika 4: Analiza radnih svojstava paralelnog SIP parsera prilikom parsiranja neispravnih SIP poruka: a) propusnost; b) razmjeran rast paralelnog SIP parsera prilikom izvođenja na četverojezgrenom procesoru u odnosu na slijedni parser

U slučaju parsiranja isključivo neispravnih poruka, radna svojstva implementacije paralelnog SIP parsera su znatno bolja (slika 4).

Naime, u tom slučaju potrebno je ispitati sve moguće puteve u sintaksnom stablu, odnosno sve nedeterminističke prijelaze u svim automatima koji parsiraju SIP poruke. Ispitivanje svih mogućih puteva odlično se može paralelizirati što potvrđuje razmjeran rast CILK implementacije parsera. Propusnost parsera u ovom slučaju je očekivano manja zbog ispitivanja svih mogućih puteva u sintaksnom stablu.

5 Zaključak

U ovome članku prikazan je dizajn paralelnog SIP parsera te načela višejezgrenih implementacija u programskom jeziku C++ pomoću CILK višejezgrene programske biblioteke i programskom jeziku Java pomoću JSR-166 višejezgrene programske biblioteke.

Eksperimentalni rezultati pokazuju uspješnu implementaciju paralelnog SIP parsera te potvrđuju ispravnost dizajna paralelnog parsera. Pri tome CILK implementacija pokazuje odličnu propusnost, dobar razmjeran rast i primjereno zauzeće memorije. Razmjeran rast je odličan u slučaju parsiranja neispravnih poruka kada se zahtijeva istraživanje svih mogućih puteva u sintaksnom stablu. JSR-166 implementacija ima u prosjeku 30% lošiju propusnost i manji faktor razmjernog rastajednim dijelom zbog toga što JSR-166 biblioteka ne koristi nikakav mehanizam za ograničavanje maksimalnog broja paralelnih procesa te raspoređivanja tih procesa na dretve operacijskog sustava.

6 Literatura

[1] J. Rosenberg et al., SIP: Session Initiation Protocol, RFC 3261, IETF, 2002.

[2] D. Crocker, P. Overell: "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, IETF, 2008.

[3] M. Cortes, J.E. Ensor, J.O. Esteban: "On SIP Performance", Bell Labs Technical Journal 9(3), Nov. 2004, p.p. 155-172

[4] S. Wanke, M. Scharf, S. Kiesel, S. Wahl: "Measurement of the SIP Parsing Performance in the SIP Express Router", Lecture Notes in Computer Science, Vol. 4606/2007, Aug. 2007, p.p. 103-110

[5] Nahum, J. Tracey, C. Wright: "Evaluating SIP Proxy Server Performance", In Proc. of the 17th International Workshop Network and Operating System Support for Digital Audio and Video(NOSSDAV), Urbana-Champaign, IL, USA, June 2007.

[6] Parsing, http://en.wikipedia.org/wiki/Parsing

[7] A. Aho, M. Lam, R. Sethi, J. Ullman: "Compilers: Principles, Techniques and Tools", 2nd Ed, Addison Wesley, 2006.

[8] "PJSIP – High Performance Open Source SIP Stack", http://www.pjsip.org/high-performance-sip.htm

[9] A. Vaha-Sipila: URLs for Telephone Calls, RFC 2806, IETF, 2000.

[10] P. Mockapetris: Domain Names - Implementation and Specification, RFC 1035, IETF, 1987.

[11] M. Sipser, Introduction to the Theory of Computation, PWS Publishing Co, 1996.

[12] T. Štefanec, I. Skuliber: "Grammar-based SIP Parser Implementation with Performance Optimizations", to be published in Proc. of ConTEL 2011 conference, Graz, Austria, June 15-17, 2011.

[13] V. Gurbani, C. Boultond, R. Sparks: "Session Initiation Protocol (SIP) Torture Test Messages for Internet Protocol Version 6 (IPv6)", RFC 5118, IETF, 2008.

[14] R. Sparks et al: "Session Initiation Protocol (SIP) Torture Test Messages", RFC 4475, IETF, 2006.

7 Popis kratica

3GPP	Third Generation Partnership Project
ABNF	Augmented Backus Naur Form
IMS	IP Multimedia Subsystem
LR parser	Left-to-right parser producing Rightmost derivation
SIP	Session Initiation Protocol

Adrese autora:

Ivan Skuliber
e-mail: ivan.skuliber@ericsson.com
Ericsson Nikola Tesla d.d.
Krapinska 45
p.p. 93
HR-10002 Zagreb
Hrvatska

Tomislav Štefanec
e-mail: tomislav.stefanec@ericsson.com
Ericsson Nikola Tesla d.d.
Krapinska 45
p.p. 93
HR-10002 Zagreb
Hrvatska

Uredništvo je primilo rukopis 24. listopada 2011.