SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 242

Radni okvir za mjerenje radnih svojstava za komunikaciju između konteksta Web preglednika

Marko Ivanković

Zagreb, lipanj 2011.

Umjesto ove stranice umetnite izvornik Vašeg rada.
Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

Hvala mojem mentoru prof. dr. Siniši Srbljiću, asistentu Ivanu Žužaku i kolegi Krešimiru Antoliću. Hvala mojim roditeljima Zdravku i Radmili. Marina, hvala ti.

SADRŽAJ

1. Uvod

U studenom 1990. Tim Berners-Lee u svoje radu "WorldWideWeb: Proposal for a HyperText Project" definira WorldWideWeb kao globalnu mrežu "HyperText stranica" (Berners-Lee i Cailliau (1990)). U izvornom prijedlogu svaka "HyperText stranica" je jedna tekstualna datoteka smještena na posebnom računalu, poslužitelju. Stranice s poslužitelja dohvaća i prikazuje poseban program koji se izvodi na klijentu, Web preglednik. Tekst sadržan u stranici može sadržavati poveznice na druge "HyperText stranice" koje se nalaze na istom ili udaljenom računalu. Termin Web stranica (*eng. Web page*), iako nije definiran u izvornom prijedlogu, s vremenom je zamijenio naziv "HyperText stranica". Termin Web mjesto (*eng. Web site*) počeo se koristiti iste godine za opisivanje skupa Web stranica koji čine logičku cjelinu. Web stranice jednog Web mjesta najčešće su na istom poslužitelju, ili istoj lokalnoj mreži poslužitelja te su održavane od iste osobe ili organizacije. S razvojem novih tehnologija kao što su Adobe Flash, JavaFX i Microsoft Silverlight pojavljuje se termin Obilne Internet Aplikacije (*eng. Rich Internet Application*). Za razliku od Web stranica, za Obilne Internet Aplikacije je potrebno nadograditi Web preglednike s proširenjima. Termin "Obilna Internet Aplikacija" prvi put se spominje 2002. godine u izvještaju tvrtke Macromedia o novoj generaciji Web stranica (Allaire (2002) i Fraternali et al. (2010)).

2004. godine W3C konzorcij zadužen za standardizaciju HyperText projekta objavio je početak rada na novom skupu standarda čija funkcionalnost bi trebala omogućiti izvedbe Obilinih Internet Aplikacija bez potrebe za proširenjima. Skup standarda je izvorno nazvan "Web aplikacije 1.0". Termin Web aplikacije s vremenom se prestao koristiti za imenovanje standarda (čije ime postaje HTML5), već se koristi za opisivanje jednog ili više Web mjesta povezanih u cjelinu koja korisniku pruža određene funkcionalnosti.

Za razliku od Web stranica za koje je definirano da se nalaze na poslužitelju, a preglednik ih samo prikazuje na klijentu, Web aplikacije se podjednako nalaze i na poslužitelju i na klijentu. Njihov izvorni kod smješten je na poslužitelju ali se izvođenje koda obavlja na klijentu u Web pregledniku. Web preglednik se iz programa za koris-

nički pristup i prikazivanje sredstvima smještenim na poslužitelju pretvara u platformu za razvoj i izvođenje složenih aplikacija. S porastom složenosti Web aplikacija javlja se potreba za međusobnom komunikacijom, kako između dijelova jedne Web aplikacije tako i između više Web aplikacija. Prve Web aplikacije s izraženom potrebom za komunikacijom su usložene primjenske stranice (*eng. mashups*), kao na primjer iGoogle stranica tvrtke Google (Srbljić et al. (2009)). Potrebu za komunikacijom ispunili su mehanizmi za komunikaciju Web aplikacija. Broj i raznolikost mehanizama za komunikaciju potaknuti su relativnom sporosti HTML5 grupe standarda u ponudi standardiziranog rješenja. Trenutni ekosustav mehanizama tako sadrži veliki broj različith rješenja. Prisutne su programske knjižnica koje komunikaciju vrše korištenjem rubnih slučajeva i iznimki prisutnim u ostalim funkcionalnostima Web preglednika. Kod novijih Web preglednika prisutni su i jednostavni mehanizmi ugrađeni u Web preglednike kao dio HTML5 standarda. Također su prisutna rješenja koja objedinjuju rubne slučajeve i HTML5 mehanizme u programske knjižnice koje olakšavaju korištenje ili nude poboljšanu funkcionalnost.

Veliki dio istraživanja vezanog uz postojeće mehanizme bavi se sigurnosnim aspektima komunikacije (Singh et al. (2010b), Barth et al. (2009)). Manji broj radova istražuje funkcionalne aspekte komunikacije (Žužak et al. (2011)), a još manji broj proučava i uspoređuje radna svojstva postojećih mehanizama. U ovom radu istražena je problematika mjerenja i usporedbe radnih svojstava mehanizama za komunikaciju. S obzirom na raznolikost i brojnost mehanizama za komunikaciju, mjerenje radnih svojstava potrebno je kako bi se dala usporedba povijesnih mehanizama sa postojećim standardom, procijenio utjecaj naprednih mogućnosti koje nude programske knjižnice, kao što je pouzdanost, na radna svojstva te ujednačilo vrednovanje različitih mehanizama kako bi se upravljalo budućim razvojem istih.

Ostatak ovog rada podijeljen je kako slijedi: u 2. poglavlju prikazan je pregled komunikacije unutar Web preglednika, svojstva sustava za komunikaciju te su opisani postojeći mehanizmi. U poglavlju 3. opisana je teorijska podloga za predložena mjerenja radnih svojstava, metodologija mjerenja te statistička obrada i prikaz rezultata. U poglavlju 4. opisani su detalji izvedbe radnog okvira dok su u 5. poglavlju prikazani rezultati mjerenja nekoliko odabranih mehanizama. U 6. poglavlje izneseni su zaključci rada te dani prijedlozi za budući rad.

2. Web preglednik i komunikacija unutar Web preglednika

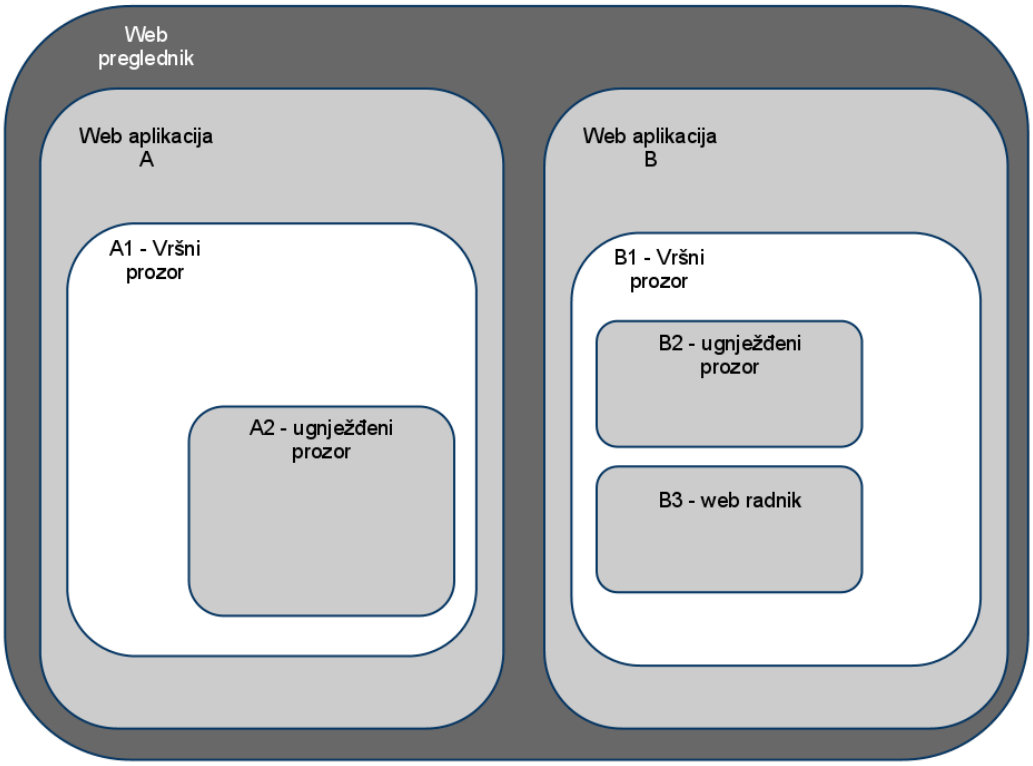
U ovom poglavlju opisan je Web preglednik kao izvršna okolina Web aplikacijama (potpoglavlje 2.1). Opisan je osnovni sigurnosni mehanizam vezan uz komunikaciju prisutan u Web pregledniku, Sigurnosna politika istog izvora (potpoglavlje 2.2). Opisan je okvir za kategorizaciju mehanizama za komunikaciju unutar Web preglednika (potpoglavlje 2.3) te su na kraju prikazani neki važniji mehanizmi za komunikaciju (potpoglavlje 2.4).

2.1. Web preglednik kao izvršna okolina Web aplikacija

Web preglednik je klijentski program sa kojim korisnik dohvaća Web stranice s Web poslužitelja. Svaka Web stranica identificirana je svojom jedinstvenom adresom (*eng. Uniform Resource Locator - URL*). Web stranica je dokument koji sadrži informacije koje prezentira korisniku, opisnike koji opisuju način prikaza informacija te programski kod pisan u skriptnom jeziku JavaScript. Programski kod neke Web stranice izvršava se u Web pregledniku te može odgovarati na korisnikove radnje, mijenjati, brisati i dodavati nove informacije te mijenjati način prikaza informacija.

Web preglednik predstavlja izvršnu okolinu programskog koda Web stranice. Nalik na mogućnosti koje klasičan operativni sustav nudi aplikacijama (Reis (2008) i Wright (2009)), Web preglednik programskom kodu unutar Web stranica nudi između ostalog mogućnost istodobnog izvršavanja više različitih Web stranica, koje ugrubo odgovaraju klasičnim dretvama, mogućnost spremanja podataka u stalnu memoriju i mogućnost komunikacije među različitim Web stranicama. Za razliku od klasičnog operativnog sustava, Web preglednik također nudi i vrlo jednostavan pristup i barata-nje s grafičkim i tekstualnim elementima Web stranice.

Arhitektura Web preglednika kao izvršne okoline Web aplikacija prikazana je na slici 2.1. Prikazan je jedan Web preglednik, koji sadrži dvije zasebne Web aplikacije



Slika 2.1: Web preglednik kao izvršna okolina

A i B. Web aplikacija A sastoji se od dva dijela A1 i A2. Web aplikacija B sastoji se od tri dijela B1, B2 i B3.

Osnovna jedinica podjele u Web pregledniku je kontekst. Kontekst se definira kao skup programskih kodova i prikazanih informacija sadržanih u nekoj Web stranici među kojima postoji točno jedna nit izvođenja. Dakle, unutar jednog konteksta nije moguće paralelno izvođenje programskog koda, dok je paralelno izvođenje više konteksta moguće. Postoje dva tipa konteksta: kontekst prozora te kontekst Web radnika (*eng. Web worker*). Kontekst prozora sadrži programski kod, informacije te izravno grafičko sučelje prema korisniku. Kontekst prozora može korisniku prikazivati informacije te od njega primati ulaz. Da bi se osigurao nesmetan rad korisnika, kontekst prozora mora ograničiti trajanje najdužeg izvršavanja programskog koda kako bi mogao reagirati na korisnički ulaz u prikladnom vremenu. Većina Web preglednika strogo ograničava trajanje izvršavanja programskog koda konteksta prozora te će sigurnosni mehanizmi zaustaviti izvršavanje i uništiti kontekst ukoliko procijeni da je trajanje prešlo neku granicu. Nasuprot prozoru, Web radnik je kontekst koji može sadržavati

samo programski kod i informacije, te nema izravno grafičko sučelje ili interkaciju s korisnikom. Zbog toga Web radnik može izvršavati programski kod neograničeno dugo. Konteksti mogu tvoriti hijerarhijsku organizaciju. Kontekst prozora može unutar sebe sadržavati drugi kontekst prozora kao ugnježđeni okvir i može sadržavati Web radnike. Web radnici mogu sadržavati samo druge Web radnike. Kontekst koji nije sadržan ni u kojem drugom kontekstu naziva se vršni kontekst. Samo prozorski kontekst može biti vršni kontekst i svaki prozorski kontekst može biti sadržan u najviše jednom kontekstu. Web radnici ne mogu biti vršni kontekst, ali mogu biti sadržani u više drugih kontekstova u kojem slučaju se nazivaju dijeljeni radnici.

Na slici 2.1 komponente A1 i B1 su vršni prozori koji sadrže komponente A2 i B2 kao ugnježđene prozore. Komponenta B1 također sadrži i Web radnika B3. Web aplikacija je dakle hijerarhijski organizirano usmjereno stablo konteksta s vršnim kontekstom koji je nužno prozor.

Svakom kontekstu pridružen je izvor, svojstvo koje opisuje adresu s koje je dohvaćena Web stranica sadržana u kontekstu. Svaka adresa sastoji se od protokola, domene, vrata, putanje, upita i fragmenta. Izvor je uređena trojka protokola, domene te vrata s kojih je sadržaj dohvaćen. Tablica 2.1 prikazuje izvore svih konteksta prisutnih na slici 2.1. Dva izvora su jednaka ako i samo ako su im u potpunosti jednaka sva tri člana.

Tablica 2.1: Izvori konteksta prikazanih na slici 2.1.

Kontekst	Protokol	Domena	Vrata
A1	HTTP	www.a.com	80
A2	HTTP	www.a.com	80
B1	HTTPS	secure.b.com	443
B2	HTTP	www.b.com	80
B3	HTTPS	secure.b.com	443

Obje komponente aplikacije A nalaze se na istom izvoru, dok se komponenta B nalazi na dva različita izvora.

2.2. Sigurnosna politika istog izvora

Web preglednici koriste izvor konteksta u velikom broju provjera prava pristupa. Vezano uz komunikaciju, najvažniji sigurnosni koncept naziva se sigurnosna politika istog izvora (*eng. same origin policy - SOP*). Politika istog izvora sprečava direktnu komunikaciju između JavaScript koda koji se izvršava unutar konteksta s različitim

izvorom, osim kod nekih rubnih uvjeta i iznimki. Politika istog izvora prvotno je obuhvaćala izravan pristup JavaScript varijablama te pristup opisniku dokumenta. S razvojem Web preglednika politika istog izvora preuzimana je za nove mehanizme kao što su XmlHttpRequest zahtjevi. Mogućnosti Web preglednika razvijene prije razvoja JavaScript jezika, kao što su učitavanje slika s različitih domena, slanje POST zahtjeva korištenjem **`<form>`** elementa te uključivanje programskog koda s različitih domena, nisu naknadno obuhvaćene politikom istog izvora (Singh et al. (2010a)).

2.2.1. Rubni slučajevi i iznimke

Politika istog izvora ne definira dobro ponašanje preglednika kod nekih rubnih slučajeva te se ono može razlikovati od Web preglednika do Web preglednika. Povijesno najznačajniji rubni slučaj odnosi se na sredstva koja nemaju dobro definirana sva tri člana izvora. Veliki broj sigurnosnih problema izazvali su protokoli **`file://`** te **`data://`** kod kojih nisu jasno definirana domena i vrata (Gollmann (2010)). Rubni slučajevi i iznimke omogućili su neke od prvih mehenizama za komunikaciju između konteksta. Prvi mehanizmi koristili su mogućnosti Web preglednika starije od politike istog izvora, kao na primjer kolačiće (*eng. cookies*), za prijenos informacija. Kako su takvi mehanizmi koristili mogućnosti koje nisu bile zamišljene za prijenos podataka, komunikacija je bila nesigurna i nepouzdana. Druga generacija mehanizama za komunikaciju koristila je iznimke posebno definirane u politici istog izvora, kao na primjer mogućnost promjene fragmenta URL adrese sa koje je učitan kontest. Mehanizmi druge generacije bolje su ostvarivali sigurnosna svojstva, ali su i dalje bili nepouzdani.

Posebna iznimka u politici istog izvora ostavljena je za domene koje dijele zajednički sufiks domene. Ukoliko oba konteksta dobrovoljno promijene svoje document.domain svojstvo na valjani zajednički sufiks izvornih domena, tada će se za sve provjere vezane uz pristup DOM modelu, ali ne i ostale radnje, primjenjivati sigurnosna ograničenja koja se primjenjuju za skripte s istog izvora. Navedena iznimka zanimljiva je zbog toga što predstavlja jedan od najranijih jasnih pokazatelja za potrebom komunikacije različitih izvora te problemima koje strogo provođenje politike istog izvora izaziva.

Politika istog izvora osim što djeluje ograničavajuće na legitimne obrasce korištenja također ne pruža potpunu zaštitu od zlonamjernog programskog koda. Kako svojstvo izvora konteksta sadrži samo protokol, domenu te vrata za potpunu zaštitu bilo bi potrebno osigurati dobronamjernost svakog programskog koda koji se nalazi na Web stranicama koje imaju jednaka ta tri člana svoje URL adrese. Kako URL

adresa sadrži još 3 člana, putanju, upit i fragment, to nije uvijek moguće. Čest slučaj kod višekorisničkih sustava je odvajanje Web stranica pojedinog korisnika korištenjem prefiksa putanje. Tako na poslužitelju s domenom `fly.fer.hr` se korištenjem protokola HTTP na vratima 80 može dohvatiti Web stranice raznih korisnika. Web stranice različitih korisnika razlikuju se po svojim prednizovima. Prefiks svih putanja Web stranica nekog korisnika obično je znak tilda ('∼') nakon kojeg slijedi korisničko ime. Tako su Web stranice s URL adresama `http://fly.fer.hr/∼marina/` i `http://fly.fer.hr/∼zlocko/` pod utjecajem različitih korisnika, međutim, JavaScript kod koji se izvodi u kontekstu učitanom s jedne može slobodno pristupati JavaScriptu koje se izvodi u kontekstu druge, zbog toga što oba konteksta imaju isti izvor.

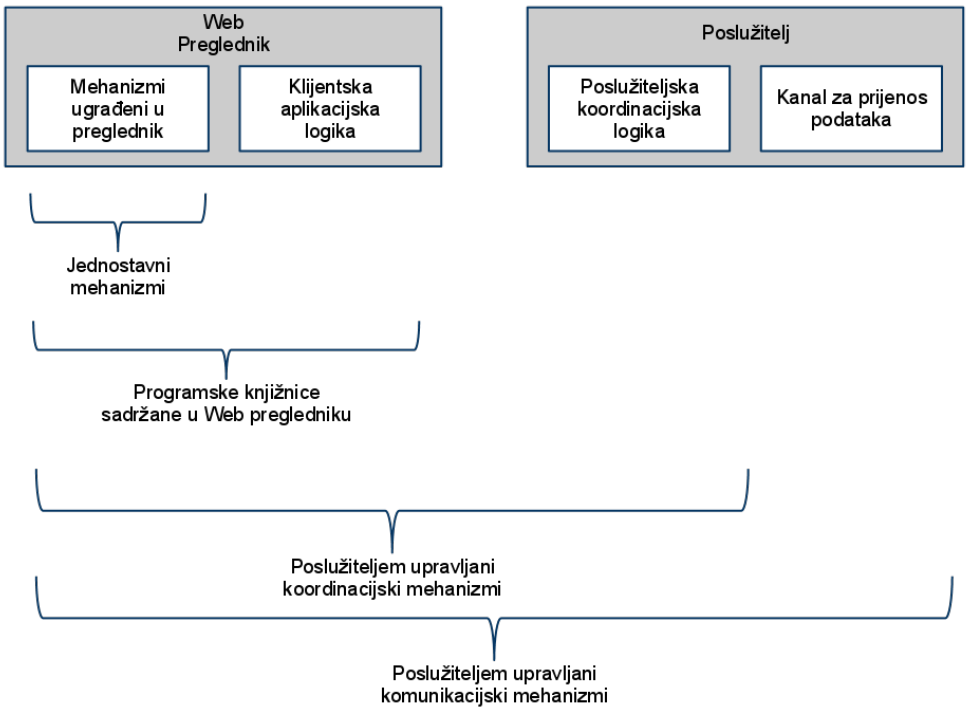
2.3. Mehanizmi za komunikaciju

Mehanizmi za komunikaciju unutar Web preglednika omogućuju razmjenu informacija između JavaScript programskog koda koji se izvršava u različitim kontekstima. Ekosustav postojećih mehanizama je vrlo opširan te se ugrubo može podijeliti na tri velike cjeline: povijesni mehanizmi koji koriste iznimke u politici istog izvora, jednostavni mehanizmi ugrađeni u Web preglednike kao dio novih standarda te programske knjižnice koje vrše nadgradnju nad jednostavnim mehanizmima i pružaju novu funkcionalnost. Mehanizme je moguće kategorizirati prema sljedećih 16 kateogrija:

1. **Tip mehanizma**

   Vrši se podjela na 4 različita tipa sustava, rastući prema dosegu izvedbe sustava. Najjednostavniji tip sustava su mehanizmi izvedeni u samom Web pregledniku. Nad njima su izgrađene programske knjižnice koji nisu izvedeni kao dio Web preglednika ali su, jednom kad su učitane kao dio neke Web stranice, u potpunosti sadržani unutar granica Web preglednika. Sljedeći po dosegu je poslužiteljem upravljani koordinacijski radni okvir. Takav sustav ovisi o udaljenom Web poslužitelju koji pruža usluge koordinacije komunikacije, iako se sam prijenos podataka i dalje odvija unutar granica Web preglednika. Najveći po dosegu izvedbe su poslužiteljem upravljani komunikacijski radni okviri, kod kojih udaljeni Web poslužitelj služi za koordinaciju i prijenos informacija. Važnost poslužiteljem upravljanih komunikacijskih radnih okvira je to što osim komunikacije između različitih konteksta unutar jednog Web preglednika, mogu služiti u svrhu komunikacije između konteksta koji se izvršavaju unutar različitih Web

preglednika.



Slika 2.2: Prikaz tipova mehanizma. Mehanizmi su prikazani rastući prema dosegu izvedbe sustava. Mehanizmi ugrađeni u Web preglednik i programske knjižnice ne prelaze granice Web preglednika, dok poslužiteljem upravljani koordinacijski i komunikacijski prelaze granicu preglednika i poslužitelja.

2. **Podrška za različite izvoree**

   Podrška za komunikaciju između različitih izvora može biti nepostojeća, u kom slučaju govorimo o komunikaciji unutar istog izvora, djelomična, koja se odnosi na komunikaciju unutar iste vršne domene, ili potpuna, u kom slučaju je komunikacija moguća između različitih konteksta bez ograničenja.

3. **Općenitost primjene**

   Većina sustava za komunikaciju generički su mehanizmi koji se mogu koristiti u svim Web aplikacijama, međutim, postoje mehanizmi ograničeni na upotrebu u specifičnim Web aplikacijama. Takvi mehanizmi često su dio specijaliziranih radnih okvira za izradu i posluživanje Web aplikacija kao što je Google Gadgets okvir.

4. **Podrška za različite vrste konteksta: Prozor i Web radnik**

   Većina mehanizama podržava komunikaciju među prozorskim kontekstima. Ma-

nji broj podržava komunikaciju među kontekstima Web radnika. Još manji broj podržava istovremeno oba tipa konteksta.

5. **Transportni mehanizam**
   Osim sustava najmanjeg dosega koji su svi mehanizmi izvedeni u samom Web pregledniku, ostali sustavi oslanjaju se na sustave nižeg tipa kao transportne mehanizme. Složeniji sustavi naslijeđuju neka svojstva svojih transportnih mehanizama.

6. **Komunikacijski model**
   Komunikacijski modeli prisutni u sustavima za komunikaciju su: model dijeljene memorije, komunikacija zasnovana na porukama, poziv udaljenih procedura te model zasnovan na pretplatama.

   Model dijeljene memorije (*eng. shared memory model*) ostvaruje razmjenu informacija čitanjem i pisanjem podataka u posebni adresni prostor radne memorije koji je pošiljatelju dostupan za pisanje, a primatelju za čitanje. Moguće je da je pošiljatelju također dostupno čitanje, ili primatelju pisanje, što olakšava upotrebu mehanizma i poboljšava radna svojstva. Nije međutim strogo nužna niti pošiljateljeva mogućnost čitanja niti primateljeva mogućnost pisanja. Primjer modela koji ne dopušta čitanje pošiljatelju je manipulacija fragmentom URL adrese.

   Model komunikacije zasnovane na porukama (*eng. Message oriented communication*) temelji se na razmjeni poruka, logičkih događaja koji sadrže informacije, između dva ili više sudionika od kojih je jedan definiran kao pošiljatelj dok su svi ostali primatelji. U tom smislu, komunikacija zasnovana na porukama je jednosmjerna u trajanju jedne poruke. Iako pošiljatelj i primatelj mogu zamijeniti mjesta, te globalno gledano komunikacija može biti dvosmjerna, u toku razmjene jedne specifične poruke komunikacija uvijek teče od pošiljatelja prema primatelju u toku cijelog prijenosa.

   Model zasnovan na pozivu udaljenih procedura (*eng. Remote Procedure Call - RPC*) temelji se na pozivanju procedura smještenih u programskom toku izvršavanja udaljenog konteksta kao da se nalaze u izvornom kontekstu. Poziv udaljenih procedura je dvosmjeran, budući da se od izvornog do udaljenog konteksta prenose parametri predani proceduri, a s udaljenog konteksta do izvora vraća povratna vrijednost izvršavanja udaljene procedure. Udaljena procedura izvršava se u memorijskom prostoru udaljenog konteksta, te može pristupati podacima u

tom kontekstu bez ograničenja.

Model zasnovan na pretplati (*eng. publish/subscribe model*) je nadogradnja modela zasnovanog na razmjeni poruka. Za razliku od modela zasnovanog na razmjeni poruka, pošiljatelj ne prenosi informacije direktno u primatelje već ih pohranjuje u središnji objekt - spremnik. Spremnik čuva poslane poruke te ih isporučuje primateljima kad ih oni zatraže. U modelu zasnovanom na pretplati može biti značajnog vremenskog odmaka u izvršavanju programskog koda pošiljatelja i primatelja. Moguće je da kontekt primatelja uopće nije prisutan u Web pregledniku u trenutku slanja, kao što i kontekst pošiljatelja može već napustiti Web preglednik u trenutku primanja poruke.

Prikladnost upotrebe pojedinog modela ovisi o složenosti i potrebama konkretne Web aplikacije.

7. **Pouzdanost**
   Vrijednosti dimenzije pouzdanosti određene su mjerom pouzdanosti komunikacije koju mehanizam pruža. Mehanizam može nuditi garanciju isporuke, pod pretpostavkom da odredište postane spremno primiti informaciju barem jednom. Mehanizam može nuditi garanciju isporuke ili greške, u kojem slučaju će poruka ili biti pouzdano isporučena, ili će se pošiljatelju pouzdano dojaviti greška. Mehanizam također može biti nepouzdan, u kojem slučaju može doći do neisporuke poruke koja može proći nezapaženo.

8. **Imenovanje**
   Programski kod smješten u neki kontekst koji želi komunicirati s programskim kodom smještenim u drugi kontekst mora moći na neki način označiti odredišni kontekst. Svojstvo imenovanja opisuje način na koji se u danom mehanizmu označava odredište. Najčešće se u mehanizmima koristi sam JavaScript objekt koji predstavlja kontekst. Druge mogućnosti su URL dokumenta koji izvršava kontekst, proizvoljni niz znakova, ili kombinacije više različitih načina.

9. **Otkrivanje**
   Usko vezano uz imenovanje, otkrivanje opisuje mogućnost mehanizma kojom programski kod može provjeriti nalazi li se imenovani kontekst unutar sustava ili ne, te dobiti referencu na njega ukoliko je potrebno.

10. **Najveća podržana veličina poruke**
    Neki mehanizmi eksplicitno ograničavaju najveću dopuštenu veličinu poruke.

Mehanizam koji ne ograničavaju eksplicitno najveću dopuštenu veličinu smatraju se neograničenim, iako su praktično ograničeni raspoloživim resursima operacijskog sustava.

11. **Povjerljivost komunikacije**
    Komunikacija je povjerljiva ukoliko samo odredišni kontekst može pristupiti prenesenim podacima.

12. **Cjelovitost komunikacije**
    Komunikacija je strogo cjelovita ukoliko podatke u prijenosu nije moguće promijeniti. Ukoliko je podatke moguće promijeniti, ali je svaku promjenu moguće uočiti, komunikacija je provjerljivo cjelovita. Ukoliko je podatke moguće promijeniti bez da se promjena uoči, cjelovitost nije podžana.

13. **Autentifikacija primatelja i pošiljatelja**
    Mehanizmi u kojima primatelj, odnosno pošiljatelj, ne može krivotvoriti entitet korišten za imenovanje unutar tog mehanizam, nude autentifikaciju primatelja odnosno pošiljatelja.

14. **Autorizacija primatelja i pošiljatelja**
    Mehanizmi koji pružaju mogućnost autorizacije primatelja pružaju mogućnost pošiljatelju da ograniči kontekste kojima se podaci smiju dostaviti. Mogućnost autorizacije pošiljatelja pruža mogućnost primatelju da navede kontekste s kojih se podaci smiju dostaviti. Način navođenja konteksta određen je svojstvom imenovanja. Mehanizam koji ne dopušta nikakvo imenovanje, to jest koji dopušta samo anonimne mehanizme, ne može nuditi ni autorizaciju.

15. **Podržanost Web preglednika**
    Podržanost sustava u većim modernim Web preglednicima. Veći preglednici su Firefox, Chrome, Opera, Safari te Internet Explorer. Razlike između inačicama Web preglednika također su istaknute kada je prikladno.

16. **Mogućnosti distribucije podataka**
    Podržava li mehanizam slanje na samo jedan kontekst (*eng. unicast*), više konteksta (*eng. multicast*) te svim kontestkima u aplikaciji (*eng. broadcast*).

Mehanizmi za komunikaciju koje razmatramo u ovom radu su prema iznesenoj kategorizaciji mehanizmi izvedeni u pregledniku ili radni okviri u potpunosti sadržani u pregledniku, s obaveznom podrškom za različite kontekste i različite izvore općenite

primjene. Ostala funkcionalna svojstva ne ograničavamo, iako njihov utjecaj na rezultate ponekad ističemo, kao na primjer kod sustava s ograničenom najvećom veličinom poruke.

2.4. Prikaz važnijih mehanizama

U ovom potpoglavlju prikazani su neki važniji mehanizmi za komunikacij u Web pregledniku. Za svaki mehanizam opisana je povijet te je prikazana klasifikacija prema 16 funkcionalnih svojstava prikazanih u potpoglavlju 2.3.

2.4.1. postMessage

Mehanizam postMessage definiran je u HTML5 skupu standarada (Hickson (2011)). Izveden je u svim najnovijim inačicama većih Web preglednika. Mehanizam postMessage izložen je kao ugrađena JavaScript funkcija te događaj. Kontekst pošiljatelj koristi ugrađenu *postMessage* funkciju kako bi zatražio od Web preglednika stvaranje *onMessage* događaja. Kontekst primatelj mora se prethodno pretplatiti na *onMessage* događaj.

Na mehanizam postMessage ne primjenjuje se politika istog izvora. Umjesto nje u standardu je definiran sigurnosni model koji omogućava autorizaciju primatelja i pošiljatelja. Pošiljatelj imenuje odredišni kontekst korištenjem JavaScript objekta koji ga predstavlja, međutim, pri slanju poruke može ograničiti domene koje su autorizirane za primanje te poruke. Web preglednik prije stvaranja *onMessage* događaja provjerava adresu pripadajuće odredišne Web stranice te je uspoređuje s zadanim regularnim izrazom koji opisuje autorizirane domene. Autorizacija pošiljatelja vrši se preko posebnog polja sadržanog u *onMessage* događaju u koje se sprema adresa pošiljatelja. Primatelj može dohvatiti adresu kao običan niz znakova te zatim na temelju te informacije postupati po želji. Najčešće primatelj provjerava odgovara li adresa pošiljatelja nekom regularnom izrazu te prema tome prima ili odbacuje poruku. Složenije odluke nisu neuobičajene, pa tako primatelj može za različite pošiljatelje imati određene različite razine dopuštenja pristupa. Web stranica nikako ne može krivotvoriti vlastitu adresu budući da Web preglednik adresu mora znati prije nego što je učitao bilo koji dio Web stranice.

Mehanizam postMessage prema klasifikaciji prikazanoj u 2.3 je jednostavni mehaizam izveden u Web pregledniku, sa potpunom podrškom za različite izvore, općenite primjene, s podrškom za oba tipa konteksta, bez posebnog transportnog mehanizma.

Komunikacijski model zasnovan mu je na razmjeni poruka te nudi nepouzdanu komunikaciju u kojoj je svaki kontekst imenovan JavaScript objektom koji ga predstavlja, ali također ima pristup adresi Web stranice učitane u kontekst. Mehanizam postMessage ne nudi otkrivanje te nema najveću veličinu poruke. Komunikacija izvedena u tom mehanizmu smatra se povjerljivom iako cjelovitost nije podržana. Konteksti su autentificirani adresama svojih Web stranica te se nad tim adresama može vršiti autorizacija pošiljatelja i primatelja. Mehanizam postMessage podržan je u svim većim Web preglednicima, iako ne u svim starijim verzijama. Mehanizam može poruke poslane s jednog konteksta dostaviti na točno jedan drugi kontekst. Slanje svim kontekstima ili istodobno slanje nekim kontekstima nije podržano.

Mehanizam postMessage važan je zbog toga što je standardizirani općeprihvaćeni mehanizam za komunikaciju unutar Web preglednika. Veliki broj naprednijih mehanizama koristi postMessage kao transportni mehanizam.

2.4.2. pmrpc

Mehanizam pmrpc, čije ime dolazi od postMessage Remote Procedure Call, složeniji je mehanizam koji koristi postMessage kao transportni mehanizam. Mehanizam pmrpc skriva mehanizam postMessage iza sloja apstrakcije koji nudi drugačiji model komuniciranja, model zasnovan na pozivu udaljenih procedura, te pouzdanost, imenovanje i otkrivanje. Konteksti koji žele komunicirati korištenjem pmrpc mehanizam moraju oba sadržavati svoju kopiju pmrpc programske knjižnice. Kontekst primatelj na početku izvođenja registrira kod pmrpc knjižnice procedure koje želi učiniti dostupnima. Kontekst pozivatelj svojoj kopiji programske knjižnice predaje referencu na kontekst te ime procedure koju želi pozvati. Kontekst pošiljatelj ime procedure mora znati unaprijed, ili ga otkriti postupkom otkrivanja kojeg pruža pmrpc. Kontekst primatelj može na kraju izvršavanja procedure izvršiti povrat vrijednosti naredbom `return`, jednako kao kod lokalnog poziva procedure. Mehanizam pmrpc automatski povratnu vrijednost prenosi kontekstu pozivatelju.

Mehanizam pmrpc prema klasifikaciji prikazanoj u 2.3 je programska knjižnica u potpunosti sadržana unutar Web preglednika, s podrškom za različite izvore općenite primjene. Mehanizam podržava oba tipa konteksta te za oba tipa koristi postMessage mehanizam kao transportni mehanizam. Za razliku od svojeg transportnog mehanizma, mehanizam pmrpc nudi Model zasnovan na pozivu udaljenih procedura. Pmrpc nudi pouzdanost u obliku garancije isporuke ili greške. U mehanizmu je podržano imenovanje i otkrivanje, a najveća podržana veličina poruke je neograničena. Komunikacija

u mehanizmu pmrpc je povjerljiva, iako nije cjelovita. Kao i kod postMessage, kontekst su autentificirani adresama svojih Web stranica te se nad njima vriši autorizacija. Mehanizam je podržan u svim većim Web preglednicima, no kontekst se sam mora pobrinuti za učitavanje programskog koda mehanizma. Model poziva udaljenih procedura omogućava pozivanje metode samo u jednom kontekstu.

Mehanizam pmrpc je uobičajen primjer nadgradnje nad postojećim mehanizmom. Mehanizam nudi veću funkcionalnost od svojeg transportnog mehanizma, no nije standardiziran te se mora učitavati u svaki kontekst koji ga želi koristiti.

3. Mjerenje radnih svojstava sustava za komunikaciju

U ovom poglavlju opisana je teorijska podloga mjerenja radnih svojstava izvedenih u radnom okviru. Željeni rezultati, mjerene vrijednosti te varijable mehanizma opisane su u poglavlju 3.1 U potpoglavlju 3.2 opisan je osnovni mjerni mehanizam, *getTime()*. U potpoglavlju 3.3 opisan je teoretski model mjrenja radnih svojstava. U potpoglavlju 3.4 opisana je statistička obrada koja se vrši nad izmjerenim vrijednostima u svrhu dobivanja konačnog rezultata. U potpoglavlju 3.4.1 opisani su načini prikaza rezultata.

3.1. Definicije varijabli, mjerenih vrijednosti i rezultata

Osnovna radna svojstva mehanizma za komunikaciju su odziv te širina prijenosa podataka. Odziv ili kašnjenje je vrijeme trajanja potpunog prijenosa jednog paketa od izvora do odredišta. Širina prijenosa podataka je najveća količina podataka koje se može prenjeti kroz mehanizam u jedinici vremena. Odziv i širina prijenosa su željeni rezultati dobiveni korištenjem radnog okvira.

Varijable mehanizma su količina informacije te broj istodobnih prijenosa informacija. Količina informacije izražava se u bajtima i predstavlja količinu korisnih informacija predanih mehanizmu. Mehanizam nad informacijama može vršiti transformacije kao što su spremanje u odgovarajuću strukturu podataka, sažimanje i slično, te ih prenosi odredištu. Utjecaj transformacija na veličinu informacija ne uzima se uobzir, već se samo provjerava identičnost dobivenih informacija na odredištu. Time se mehanizmima koji vrše transformacije u svrhu povećanja radnih svojstava uzimaju u obzir sva poboljšanja ili pogoršanja nastala iz tih transformacija. Očekivano ponašanje za većinu mehanizama je povećanje odziva s porastom količine informacija. Kod rijetkih mehanizama moguća je stagnacija za neke vrijednosti. Također je očekivan nagli

pad radnih svojstava ili općeniti prestanak rada sustava kod gornje granične količine informacija. Teoretska gornja granica uvijek je ukupni raspoloživi memorijski prostor računala. Praktična gornja granica ovisi o samom mehanizmu i njegovom baratanju memorijskim prostorom. Istodobni prijenosi informacija mogući su kod nekih mehanizama. Kod takvih mehanizama broj istodobnih prijenosa može utjecati na radna svojstva. Rubni slučaj istodobnih prijenosa je 1 prijenos, kojeg se uvijek razmatra posebno. Kod mehanizma koji ga podržavaju, istodobni prijenos može povećati širinu prijenosa. S porastom broja istodobnih prijenosa može doći do zagušenja sustava, što ima drastičan utjecaj na radna svojstva.

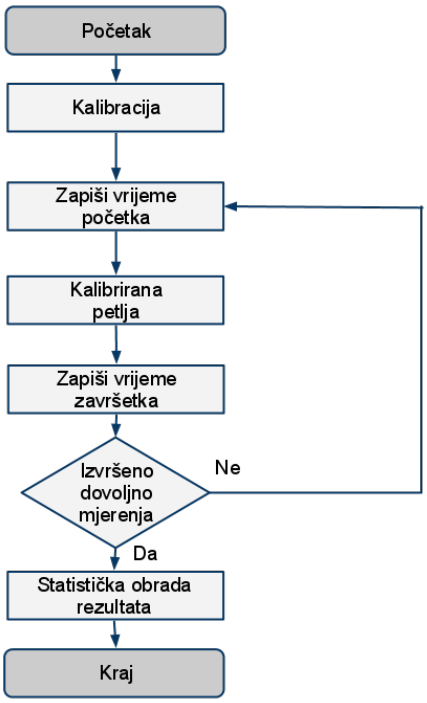
Izvorne mjerene vrijednosti su vrijeme trajanja potpunog prijenosa jednog paketa informacija od izvora do odredišta te ukupno vrijeme potrebno za prijenos svih informacija. Ukupno vrijeme od posebne je važnosti u slučaju istodobnog prijenosa informacija.

3.2. Osnovni mjerni mehanizam

Mjerenje vremena u izvedenom radnom okviru vrši se korištenjem *getTime()* naredbe ugrađene u JavaScript jezik. Korištenjem naredbe *getTime()* postiže se prenosivost radnog okvira među raznim Web preglednicima i operativnim sustavima budući da je ona univerzalno dostupna. Kod osnovnog mjernog mehanizma potrebno je pokazati točnost i preciznost. Točnost definiramo kao odstupanje srednje vrijednosti većeg broja mjerenja od referentne vrijednosti, a preciznost kao širinu osipanja većeg broja mjerenja oko referentne vrijednosti. Točnost i preciznost *getTime()* naredbe nije istražena u prijašnjim radovima. Neformalni radovi u kojima se analizira preciznost su Resig (2008) te Ivanković (2011). Točnost *getTime()* naredbe teško je pokazati u automatiziranom eksperimentu zbog potrebe za vanjskim vrlo preciznim izvorom vremenskih informacija. Teoretska najveća točnost *getTime()* naredbe ograničena je najvećom točnosti operativnog sustava na kojem se Web preglednik izvršava te povratnom vrijednosti u samom Web pregledniku. Jezgra operativnog sustava Linux u teoriji nudi točnost reda veličine nanosekunde iako u praksi tu točnost postiže samo na specijaliziranim računalima. Na prosječnom računalu točnost reda veličine mikrosekunde je uobičajena. Točnost jezgre operativnog sustava Windows uvelike ovisi o verziji operativnog sustava i u najgorem slučaju je reda veličine 10 milisekundi (Berger (2003)). U Web preglednicima Firefox, Chrome, Safari te Internet Explorer *getTime()* naredba definirana je kao broj milisekundi proteklih od UNIX epohe, te je stoga implicitno točnost ograničena na red veličine milisekunde.

3.2.1. Preciznost *getTime()* naredbe

Preciznost *getTime()* naredbe pokazana je eksperimentalno. Na slici 3.1 prikazan je dijagram toka eksperimenta. Glavni dio eksperimenta je središnja petlja koja simulira



Slika 3.1: Scenarij mjerenja preciznosti *getTime()* naredbe

izvršavanje nekog programskog odsječka čije trajanje mjerimo. Središnja petlja kalibrira se u prvom koraku eksperimenta kako bi njezino trajanje bilo blizu nekoj zadanoj vrijednosti. Kako se za tu kalibraciju koristi *getTime()* naredba moguće su znatne razlike između kalibrirane i zadane vrijednosti, međutim, kako se kalibracija koristi za postizanje dovoljnog trajanja i smanjivanje utjecaja različitih brzina procesorskih arhitektura sama točnost kalibracije ne utječe na rezultat. U praksi se pokazuje da je kalibracija točna do u red veličine, što je za potrebe mjerenja dovoljno. Nakon kalibracije se nekoliko puta izvodi mjerenje. Jedno mjerenje sastoji se od zabilješke vremena početka, izvršavanja, jednog izvršavanja kalibrirane petlje te zabilješke vremena završetka. Broj ponavljanja mjerenja mora biti dovoljno velik kako bi se statistički značajno moglo zaključiti ponašanje standardne devijacije koja je najvažniji rezultat statističke obrade, zajedno uz oblik raspodjele rezultata. Ukoliko oblik raspodjele odgovara Gaussovoj raspodjeli, *getTime()* naredba je precizna. Preciznost na dva različita preglednika ili operativna sustava može se relativno usporediti korištenjem standardnih devijacija. Manja standardna devijacija indicira precizniju izvedbu *getTime()* naredbe.

U programskoj implementaciji eksperimenta petlja je implementirana kao kod vremenske složenosti O(n^2). Kvadratna složenost odabrana je kako bi se omogućilo bolje upravljanje vremenom izvođenja. Kalibracija se vršila na okvirnih 75 milisekundi, a broj uzoraka na 500. Tablica 3.1 prikazuje sumirane rezultate svih eksperimenata mjerenja preciznosti *getTime()* naredbe. Iz tablice je vidljivo da je na operativnom sustavu

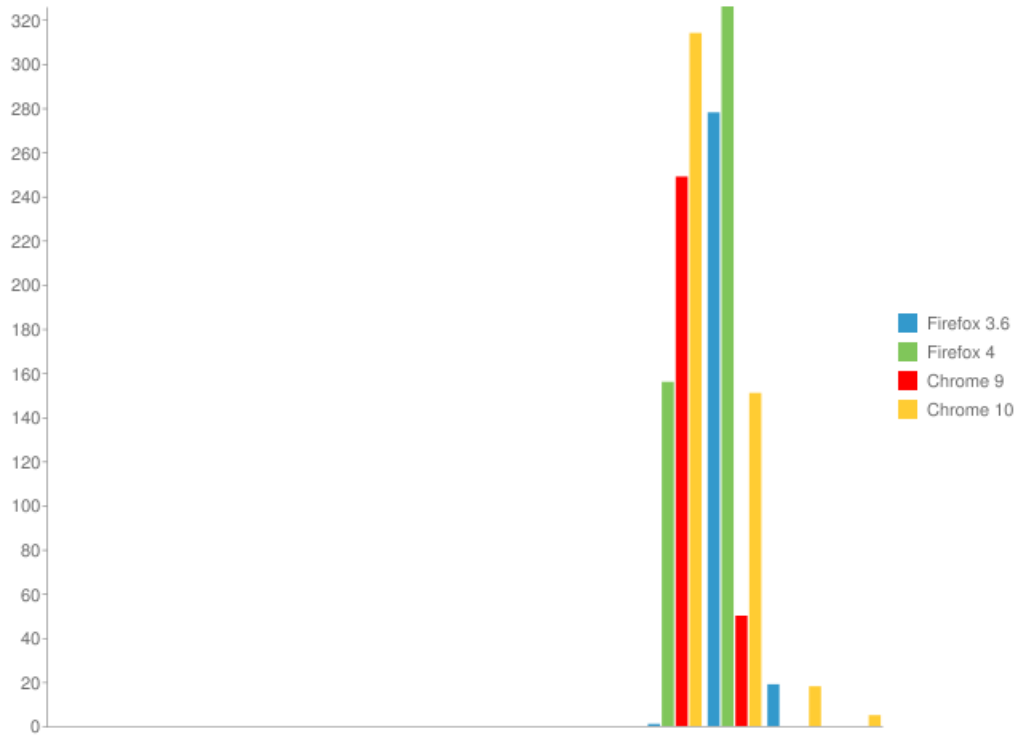
Tablica 3.1: Sumirani rezultati eksperimenta mjerenja preciznosti *getTime()* naredbe.

	Linux	Windows XP	Windows 7
Firefox 3.6	0.5ms	6.8ms	[1.2ms, 7.3ms]
Firefox 4.0	0.4ms	37.4ms	[3.6ms, 7.9ms]
Chrome 9	1.2ms	12.7ms	[4.7ms, 17.3ms]
Chrome 10	1.7ms	33.8ms	[4.9ms, 19.7ms]

Linux preciznost reda veličine 1 milisekunde što je prihvatljivo za mjerenja. Mjerenja su provedena na većem broju različitih računala kako bi se utvrdio utjecaj sklopovlja na rezultate. Na operativnom sustavu Linux nije bilo utjecaja. Na slici 3.2 prikazana su mjerenja za sve preglednike na operativnom sustavu Linux. Vidi se vrlo usko grupiranje rezultata. Na operativnom sustavu Windows 7 bilo je značajnih varijacija te su rezultati dani kao interval najmanje i najveće viđene standardne devijacije. U najboljem slučaju rezultati na operativnom sustavu Windows 7 su tek neznatno lošiji od operativnog sustava Linux. U najgorem slučaju su red veličine lošiji. Samo variranje preciznosti čini svako mjerenje izvedeno na Windows 7 operativnom sustavu okvirnim te će se u ovom radu rezultati dobiveni na tom operativnom sustavu davati samo okvirno u svrhu potpunosti. Operativni sustav Windows XP je nedovoljno precizan te se također neće uključivati u daljnja mjerenja, osim okvirno. Na slici 3.3 prikazana je jedna uobičajena distribucija rezultata na operativnom sustavu Windows.

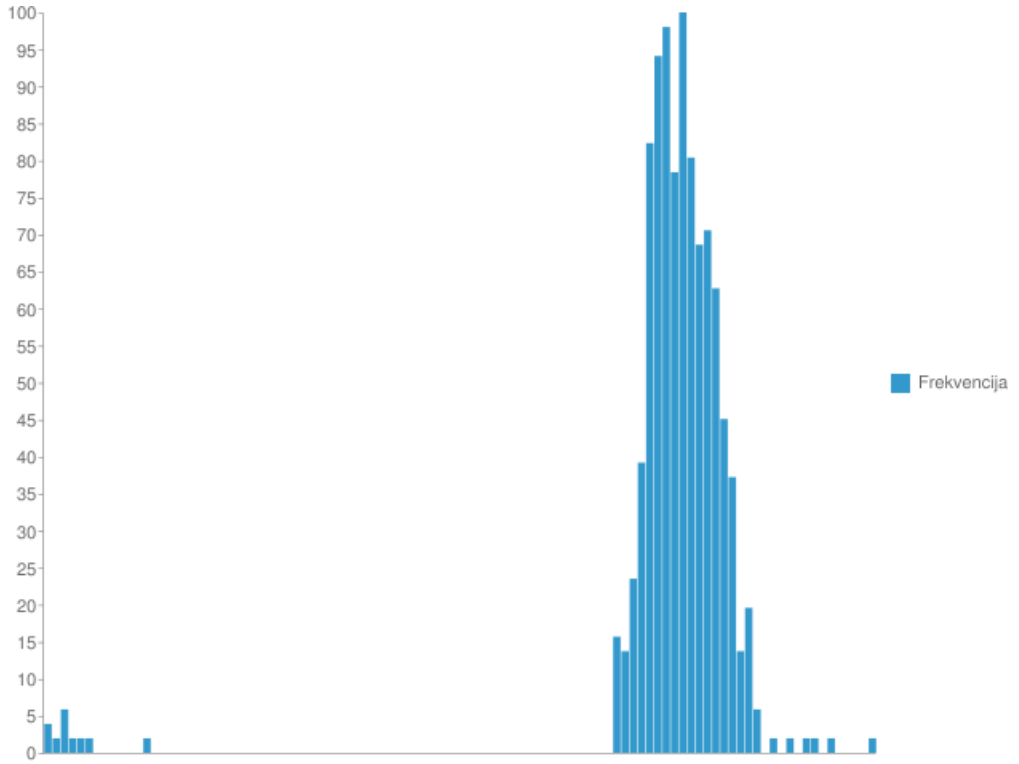
3.3. Metodologija mjerenja

Osnovni scenarij nad kojim se vrši mjerenje radnih svojstava prikazan je na 3.4. Scenarij sadrži četiri komponente: Kontekst A, CCCMBM A, CCCMBM B i Kontekst B. Kontekst A je "glavni" kontekst koji započinje mjerenje. CCCMBM A i B su kopije radnog okvira učitane u kontekste. Kontekst B je "sporedni" kontekst koji samo preusmjerava poruke. Na početku mjerenja oba konteksta se registriraju kod svoje kopije radnog okvira. Registracijom se radnom okviru omogućava korištenje mehanizma razmjene informacija sadržanog u kontekstima. Kontekst A pokreće test, nakon



Slika 3.2: Rezultati mjerenja preciznosti *getTime()* naredbe na operativnom sustavu Linux. Na osi ordinata nanesene su vrijednosti broja pojavljivanja pojedine vrijednosti apcise u svim mjerenjima. Na osi apcisa nanesena je normalizirana vrijednost jednog mjerenja. Rezultati su normalizirani tako da u svakom nizu rezultata pronađen najmanji broj, koji je zatim oduzet od svih brojeva u nizu. Nakon toga je svim brojevima pribrojena odabrana konstanta.

čega CCCMBM A priprema informacije, bilježi vrijeme predaje poruke kontekstu A i predaje poruku kontekstu A. Kontekst A zatim koristi svoj mehanizam prenosa informacija te ih prenosi kontekstu B koji ih predaje CCCMBM B. CCCMBM B bilježi vrijeme primitka poruke, koje zatim šalje kroz kontekst B i kontekst A natrag do CCCMBM A. CCCMBM A sad može izračunati razliku oba zabilježena vremena. U toku osnovnog scenarija mjeri se trajanje točno jedne izmjene informacija među kontekstima. Kako bi se omogućila statistička obrada rezultata, osnovni scenarij je potrebno ponoviti više puta. Ponavljanje osnovnog scenarija moguće je izvršiti na dva načina: uzastopno (sinkrono) i istodobno (asinkrono). Kod uzastopnog ponavljanja CCCMBM A prije svakog slanja poruke čeka dolazak potvrde za prethodno poslanu poruku. Kod istodobnog ponavljanja CCCMBM A šalje poruke najvećom brzinom kojom može. Rezultat uzastopnog ponavljanja je prosječno vrijeme odaziva sustava dok je rezultat istodobnog ponavljanja širina prijenosa podataka. Količina informacija prenesenih u jednom mjerenju je varijabla mjerenja. Najmanja količina informacija je 1B, što pred-



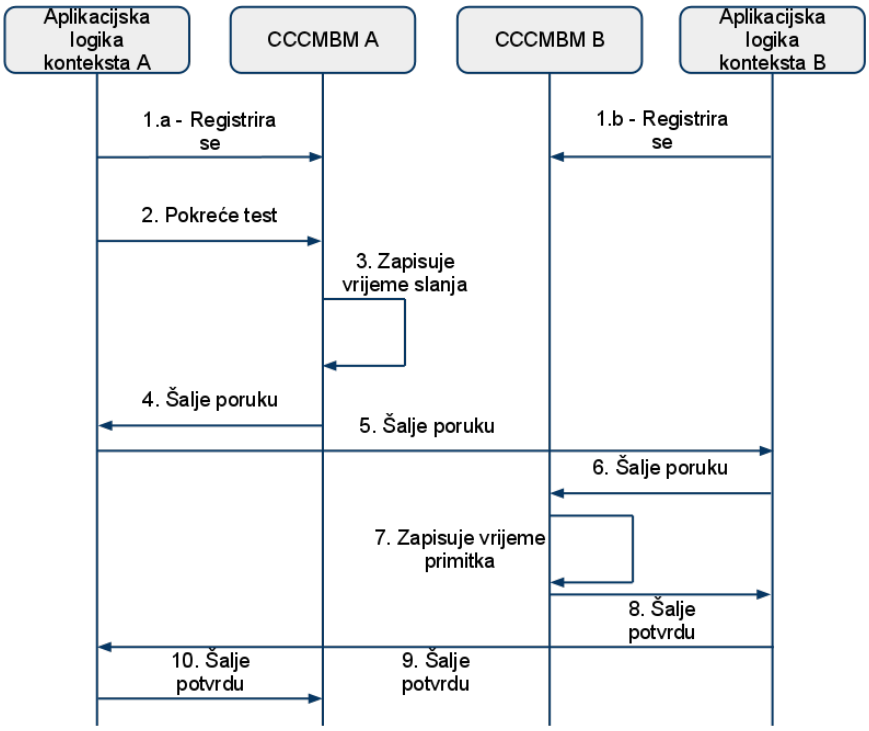
Slika 3.3: Rezultati mjerenja preciznosti *getTime()* naredbe na operativnom sustavu Windows. Na osi apcisa nalaze se vrijednosti mjerenja trajanja jednog koraka mjerenja. Na osi ordinata nazali se broj pojavljivanja odgovarajuće vrijednosti apcise u svim mjerenjima.

stavlja slanje poruke koja sadrži samo redni broj poruke. Najveća količina informacija može biti eksplicitno ograničena od strane komunikacijskog mehanizma ili implicitno ograničenjem operativnog sustava i računalnog sklopovlja. Usporedba više različitih mehanizama uvijek se vrši do najveće moguće veličine podržane od strane svih mehanizama koje se uspoređuje.

3.4. Statistička obrada rezultata

Prikupljeni podaci u svojem osnovnom obliku su niz uređenih parova cijelih brojeva koji predstavljaju vrijeme izlaska poruke iz CCCMBM A i ulaska u CCCMBM B u milisekundama. Nazovimo te nizove A i B, a broj elemenata u svakom nizu označimo sa n.

$$n = \text{broj poslanih/primljenih poruka}$$
$$A = \{slanje_1, slanje_2, \ldots, slanje_n\}$$
$$B = \{primitak_1, primitak_2, \ldots, primitak_n\}$$



Slika 3.4: Osnovni scenarija mjerenja. Prikazana su dva konteksta koji mogu ali nemoraju imati isto svojstvo izvora. Svaki kontekst sastoji se od dva dijela, radnog okvira te aplikacijske logike.

Prvi korak obrade je stvaranje novog niza, D takvog da je svaki element u nizu razlika vremena slanja i vremena primanja pojedine poruke.

$$D_i = B_i - A_i$$

To je dakle ukupno trajanje slanja poruke zajedno s prebacivanjem poruke iz CCCMBM u kontest te obratno. Zatim se izračunava srednja vrijednost niza D.

$$\bar{d} = \frac{1}{n}\sum_{i=1}^{n} D_i$$

Nakon srednje vrijednosti uzoraka izračunava se procjena standardne devijacije uzorka s Besselovom korekcijom.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\bar{d} - d_i)^2}$$

Dobiveni s je procjena prave standardne devijacije trajanja slanja. Pomoću procjenjene standardne devijacije i srednje vrijednosti uzorka procijenjujemo 95% interval

pouzdanosti.

$$(\bar{d} - 1.96\frac{s}{\sqrt{n}} \leq d \leq \bar{d} + 1.96\frac{s}{\sqrt{n}})$$

Interval pouzdanosti zajedno sa srednjom vrijednosti je završni rezultat statističke obrade rezultata mjerenja. Usporedba dva mjerenja može se vršiti na temelju intervala. Ukoliko je veća granica intervala X manja od manje granice intervala Y kažemo da je metoda komunikacije mjerena u X brža od metode komunikacije mjerene u Y. Ukoliko je manja granica intervala X veća od veće granice intervala Y kažemo da je metoda komunikacije mjerena u X sporija od metode komunikacije mjerene u Y. U svim ostalim slučajevima nemamo dovoljno podataka za usporedbu metoda X i Y te definiramo razliku brzine metoda X i Y kao nerazlučivu danim podacima. Srednja vrijednost intervala koristi se za usporedbu mjerenja koja se vrši na istoj metodi komunikacije uz različite parametre sustava. Ukoliko se radi o istodobnom mjerenju, dobiveni rezultat još naknadno normaliziramo tako da količinu prenesenih informacija podijelimo s oba kraja intervala. Time dobivamo širinu prijenosa podataka izraženu u bajtovima po milisekundi. Kako bi rezultati bili primjereniji svakodnevnoj upotrebi, prema potrebi vršimo pretvorbu mjernih jedinica u bajtove po sekundi, kilobajte po sekundi, megabajte po sekundi ili gigabajte po sekundi.

3.4.1. Prikaz rezultata

Prikaz rezultata u radnom okviru mora ispunjavati sljedeća svojstva:

1. Jednostavnu usporedbu dva mehanizma
2. Jednostavni pregled ukupnih rezultata svih mjerenja za pojedini mehanizam
3. Jednostavni pregled rezultata jednog mjerenja pojedinog mehanizmau
4. Jednostavna usporedba rezultata više mjerenja istog tipa, ali različitih ulaznih varijabli, jednog mehanizma
5. Jednostavna usporedba rezultata više mjerenja istog tipa i istih vrijednosti varijabli, više različitih mehanizama
6. Jednostavna usporedba rezultata više mjerenja istog tipa, različitih ulaznih varijabli i više različitih mehanizama

Za postizanje navedenih svojstava koriste se dva načina prikazivanja: tablični i grafički. Tablični prikaz osigurava prvo svojstvo, a grafički prikaz osigurava ostala svojstva. Tablični prikaz osigurava prvo svojstvo zbog toga što su dva broja dovoljna za

ukupni opis mehanizma, najmanji odziv i najveća širina prijenosa. Za usporedbu dva mehanizma dovoljno je dakle izvršiti dvije usporedbe decimalnih brojeva. Grafički prikaz prikladan je za ostale preglede rezultata zbog toga što se svi rezultati mogu opisati nizom uređenih parova decimalnih brojeva. Nizovi uređenih parova prirodno se prikazuju u koordinatnoj ravnini.

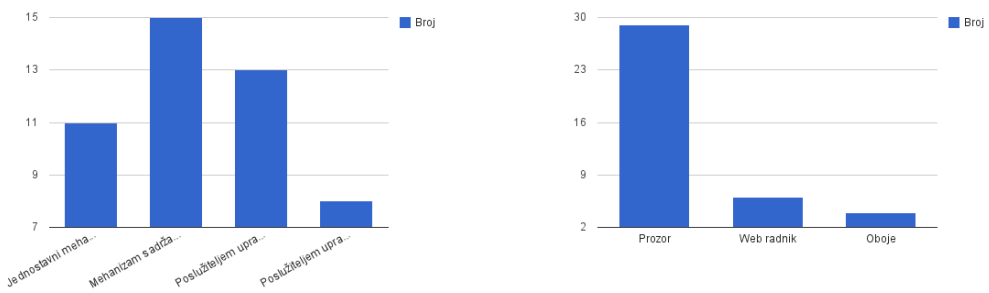
4. Radni okvir

U ovom poglavlju iznesen je opis arhitekture izvedenog radnog okvira za mjerenje radnih svojstava komunikacijskih mehanizama. Motivacija za izradu radnog okvira opisana je u potpoglavlju 4.1. Zahtjevi i scenariji upotrebe radnog okvira opisani su u potpoglavlju 4.2. Podpoglavlje 4.3 opisuje arhitekturu izvedenog radnog okvira. Sučelje prema korisnicima radnog okvira opisano je u potpoglavlju 4.4. Način izvedbe statističke obrade rezultata te prikaza rezultata opisani su u potpoglavlju 4.5. Način na koji radni okvir upravlja tokom mjerenja opisan je u potpoglavlju 4.6. U potpoglavlju 4.7 prikazani su primjeri korištenja radnog okvira.

4.1. Motivacija

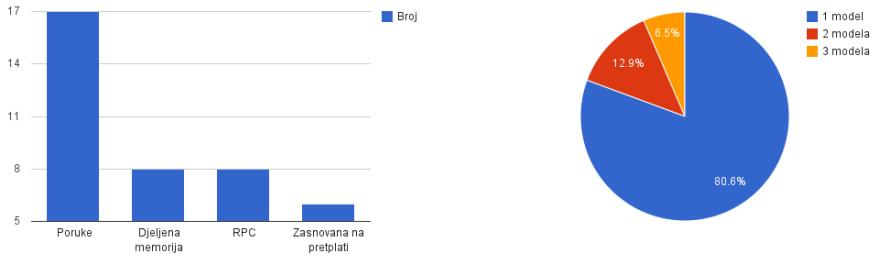
Izrada radnog okvira za mjerenje radnih svojstava motivirana je raznolikošću ekosustava mehanizma za komunikaciju. Prema dimenzijama izloženim u 2.3 klasificirano je 30-tak važnijih mehanizma za komunikaciju u Web pregledniku. Na slikama 4.1, 4.2 i 4.3 prikazani su rezultati klasifikacije.

Na slici 4.1 prikazani su rezultati podjele postojećih mehanizama po tipu mehanizma te podržanom tipu konteksta. 11 mehanizama su jednostavni mehanizmi ugra-



(a) Klasifikacija po tipu mehanizma **(b)** Klasifikacija po podržanom tipu konteksta

Slika 4.1: Rezlutati klasifikacije po tipu mehanizma i tipu podržanih konteksta.



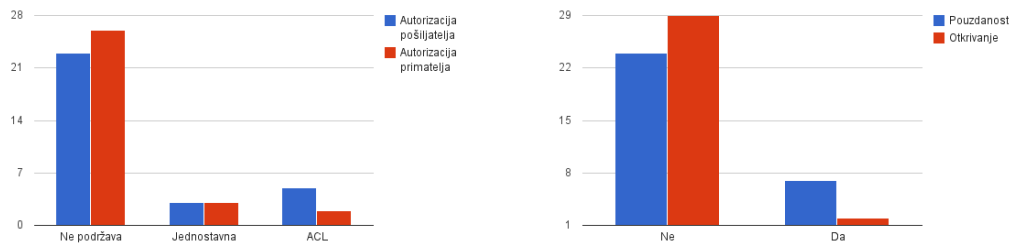
(a) Podržani komunikacijski modeli **(b)** Broj podržanih komunikacijskih modela

Slika 4.2: Rezlutati klasifikacije po podržanim komunikacijskim modelima

đeni u Web preglednik, 15 mehanizama mogu biti u potpunosti sadržani u Web pregledniku, 13 mehanizama može se koristiti poslužiteljem za upravljanje koordinacijom, a 8 a upravljanje komunikacijom. Neki mehanizmi pripadaju u više različitih tipova, ovisno o mogućnostima koje se koriste. Na primjer neke programske knjižnice u Web preglednicima koji podržavaju postMessage mehanizam su u potpunosti sadržane u Web pregledniku, ali u Web preglednicima koji ne podržavaju postMessage su poslužiteljem upravljani koordinacijski. Po tipu konteksta, 29 mehanizama podržava samo prozorski tip konteksta, 6 mehanizama podržava samo Web radnike a samo 4 mehanizma podržavaju oba tipa konteksta.

Na slici 4.2 prikazani su rezultati podjele postojećih mehanizama po podržanim komunikacijskim modelima. 17 mehanizama podržavaju komunikaciju zasnovanu na porukama, 8 mehanizama podržava dijeljenu memoriju, 8 mehanizama pozive udaljenih procedura i 6 mehanizama komunikaciju zasnovanu na pretplati. Neki mehanizmi podržavaju više od jednog komunikacijskog modela, iako su rijetki. 25 mehanizama (80%) podržava samo jedan komunikacijski model, 4 mehanizma (13%) podržava 2 modela, a samo 2 mehanizma (6%) podržava 3 modela. Niti jedan mehanizam ne podržava sva 4 modela.

Na slici 4.3 prikazani su rezultati podjele mehanizama po podršci za autorizaciju, imenovanje i pouzdanost. Mehanizam može podržavati autorizaciju pošiljatelja, autorizaciju primatelja ili oboje. 23 mehanizma ne podržavaju nikakvu autorizaciju pošiljatelja, 26 ne podržavaju nikakvu autorizaciju primatelja. 3 mehanizma podržavaju jednostavnu autorizaciju primatelja i pošiljatelja. Mehanizam koji nudi jednostavnu autorizaciju omogućava korisniku imenovanje jednog entiteta koji ima potpuni pristup, a svi ostali entiteti nemaju nikakav pristup. Za razliku od toga, mehanizmi koji nude upravljanje listama pristupa (*eng. Access Control List - ACL*) omogućavaju korisniku



(a) Podrška autorizacije

(b) Podrška imenovanju i pouzdanosti

Slika 4.3: Rezlutati klasifikacije po podršci autorizaciji, imenovanju te pouzdanosti

imenovanje više različitih entiteta s različitim razinama pristupa. ACL autorizaciju pošiljatelja podržava 5 mehanizama, a autorizaciju primatelja podržavaju samo 2 mehanizma. Pouzdanost garantira samo 7 mehanizama. Samo 2 mehanizma podržavaju otkrivanje.

Usporedba radnih svojstava osnovni je dio usporedbe različitih mehanizama. Kako bi se osigurala smislenost usporedbe, vrlo je važno osigurati jednakost mjerenja za različite mehanizme. S obzirom na prikazanu heterogenost mehanizama, radni okvir je jedini način na koji se jednakost mjerenja može osigurati. Složenost problema mjerenja vezana uz heterogenost ekosustava mehanizama povećava se za red veličine sa uvođenjem heterogenih Web preglednika. Radni okvir omogućuje centraliziranu integraciju s različitim Web preglednicima te laku prenosivost mjerenja među Web preglednicima.

Prikazana heterogenost motivacija je za izradu radnog okvira, ali je i uzrok većine izazova kod izrade.

4.2. Scenariji upotrebe i zahtjevi

Radni okvir mora pružati sljedeće mogućnosti:

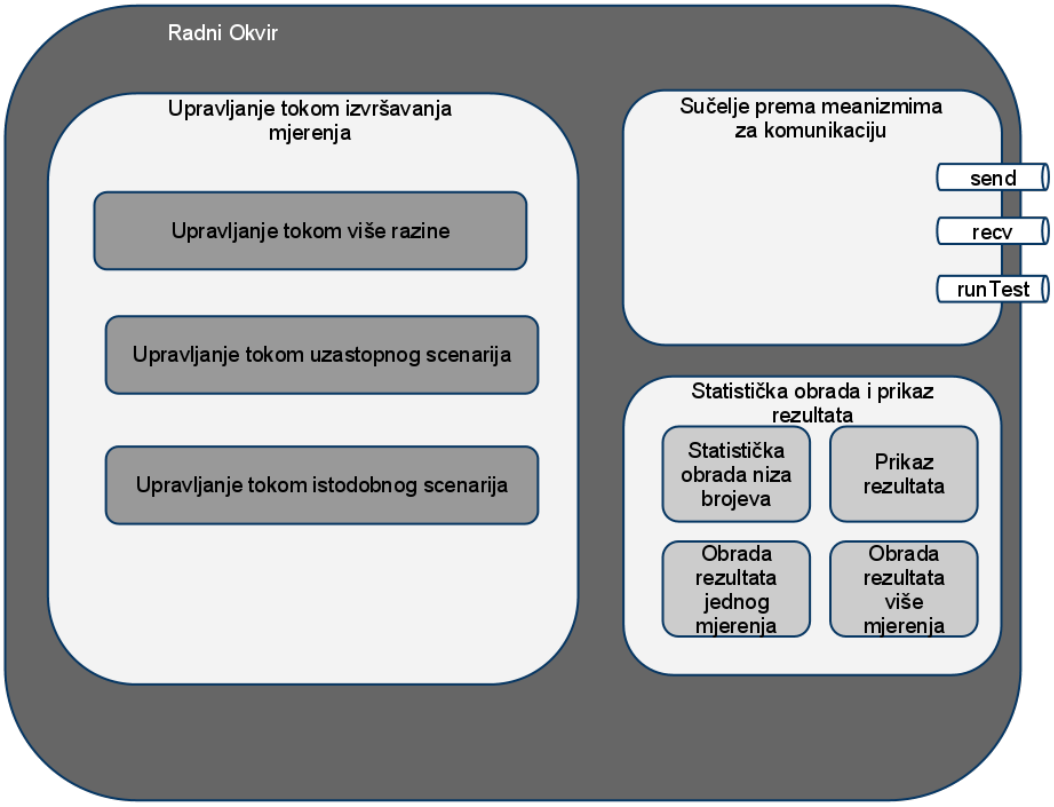
1. Prenosivost među Web preglednicima
2. Podrška za komunikacijske modele zasnovane na porukama, pozivu udaljenih procedura te dijeljenoj memoriji
3. Podrška za mehanizme u potpunosti sadržane u Web pregledniku

Kako bi se ostvarila prenosivost među različitim Web preglednicima radni okvir izvedne je u postpunosti u skriptnom jeziku JavaScript. Radni okvir koristi samo mate-

matičke operacije i osnovnu funkcionalnost prisutnu u svim većim Web preglednicima. Budući da je radni okvir sadržan u potpunosti u Web pregledniku, podržani su mehanizmi ugrađeni u sam Web preglednik, mehanizmi u potpunosti sadržani u Web pregledniku te poslužiteljem upravljani koordinacijski mehanizmi. Radni okvir moguće je teoretski koristiti i kod nekih poslužiteljem upravljanih komunikacijskih mehanizma, ali je važno izvršavati komunikaciju unutar jednog Web preglednika ili riješiti problem sinkronizacije trenutnog vremena na više računala. Poneki mehanizmi specijalizirane primjene mogli bi imati problem s korištenjem radnog okvira, ukoliko njihovo područje primjene zabranjuje korištenje proizvoljnog JavaScript koda, na primjer, korištenjem Caja sigurnosnog okruženja. Mehanizmi kod kojih je ograničena veličina poruke mogu koristiti radni okvir, iako su njihovi rezultati ograničeno usporedivi s ostalima, osim u ekstremnim slučajevima najveće veličine poruke od nekoliko bajtova. Radnom okviru potrebno je barem 12 bajtova za prenos kontrolnih informacija (4 za identifikacijski broj poruke te 8 za vrijeme u milisekundama). Radni okvir koristi apstraktne pojmove "slanja" i "primanja" čija konkretna implementacija je prepuštena mehanizmu za komunikaciju. Time se postiže agnostičnost po pitanju podržanog komunikacijskog modela. Radni okvir podržava komunikacijski model zasnovan na porukama, pozivu udaljenih procedura te dijeljenoj memoriji. Komunikacijski model zasnovan na pretplati također je podržan, no interpretacija rezultata mora uzeti u obzir specifičnosti tog modela. Kod komunikacijskog modela zasnovanog na pretplati nužno postoji posrednik, u obliku niza poruka ili usluge, koji prima informacije od pošiljatelja, te zatim kao drugu operaciju šalje informaciju primatelju. Kod svih ostalih modela razmjena informacija je atomarna, dok se ovdje sastoji od dvije operacije koje mogu biti odmaknute u vremenu.

4.3. Arhitektura radnog okvira

Arhitektura radnog okvira prikazana je na slici 4.4. Radni okvir podjeljen je u tri logičke cjeline: sučelje prema mehanizmu komunikacije, upravljanje tokom izvršavanja mjerenja te statistička obrada i prikaz rezultata. Sučelje prema mehanizmu komunikacije je najmanji dio i sadrži samo tri funkcije: send, recv i runTest. Funkcije send i recv su sučelje prema mehanizmu komunikacije u strogo programskom smislu objektnog oblikovanja. Od mehanizma se očekuje da ih prije pokretanja mjerenja pozivom funkcije runTest prepiše sa odsječcima koda koji će primati i slati poruke na način specifičan za taj mehanizam. Radni okvir obavlja statističku obradu rezultata izravnim korištenjem matematičkih metoda prikazanih u 3.4. Prikaz rezultata vrši se ispisom



Slika 4.4: Arhitektura radnog okvira. Na slici je prikazna logička podjela radnog okvira na komponente. Svaka komponenta obavlja jednu funkciju u radnom okviru koja je naznačena u imenu komponente.

HTML tablica te korištenjem Google Chart Server servisa koji omogućava dinamičku izradu grafičkog prikaza rezultata u obliku stupčastih i linijskih grafova. Zadnja cjelina, upravljanje tokom izvršavanja mjerenja, odgovorna je za izvođenje eksperimenta. Upravljanje tokom izvršavanja mjerenja osigurava da se u toku uzastopnog mjerenja poruke prenose uzastopno, bira broj ponavljanja jednog mjerenja, veličinu, broj i sadržaj svake poruke te se brine za prenos i usklađivanje informacija izvršnih tokova radnog okvira u svim kontekstima u kojima se odvija mjerenje.

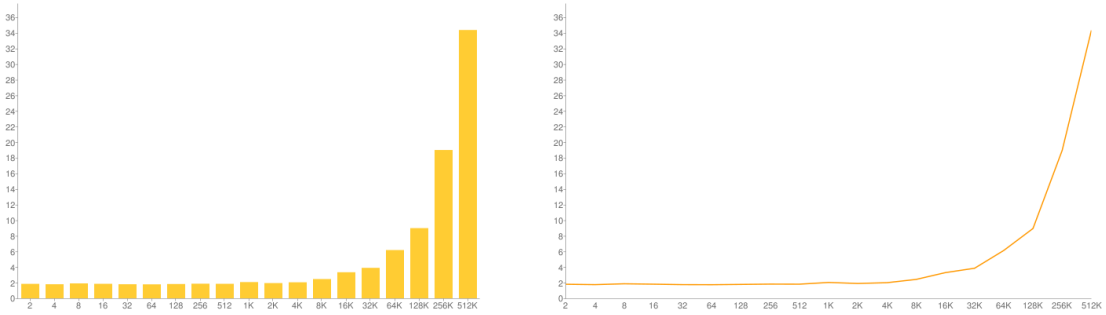
4.4. Sučelje prema komunikacijskim mehanizmima

Sučelje prema komunikacijskom mehanizmu sastoji se od 3 funkcije: send, recv i runTests. Funkcija runTests poziva se jednom, na početku mjerenja te pokreće mehanizme upravljanja tokom mjerenja. Funkcije send i recv su prazne funkcije koje pojedini mehanizam mora zamijeniti sa vlastitom implementacijom. Funkcija send mora primiti dva parametra: identifikacijsku oznaku poruke i sadržaj poruke, te ih poslati na odre-

dišni kontekst na način koji je smislen za odabrani mehanizam. Funkcija recv mora zaprimiti poruku koju je poslala funkcija send korištenjem odabranog mehanizma te zatim izvršiti potrebnu obradu kojom iz poruke čita identifikacijsku oznaku i sadržaj. Na kraju funkcija recv mora pozvati posebnu recvRet funckiju kojoj predaje identifikacijsku oznaku i sadržaj.

4.5. Prikaz i statistička obrada rezultata

Prva statistička obrada vrši se na kraju svakog mjerenja. Obrada je implementirana direktnom izvedbom foruma prikazanih u 3.4. Ulazni podaci za statističku obradu su sadržani u strukturi podataka koja za svaku identifikacijsku oznaku poslane poruke sadrži: oznaku poruke, vrijeme slanja, vrijeme prijema te sadržaj poruke. Izlazni podaci iz statističke obrade sadržani su u posebnoj strukturi podataka koja sadrži srednju vrijednost trajanja prenosa poruka, procjenu standardne devijacije, 95% interval pouzdanosti prosječnog trajanja prijenosa te niz u kojem su zapisane frekvencije pojavljivanja svakog trajanja. Dobivena struktura podataka proslijeđuje se funkcijama za tablični i grafički prikaz rezultata. Po završetku jednog skupa mjerenja, na primjer više mjerenja istog mehanizma s različitim veličinama poruka, podaci dobiveni pojedinim statističkim analizama prosljeđuju se modulu za drugu statističku obradu. Druga statistička obrada izračunava razlike između pojedinih mjerenja, frekvenciju pojavljivanja pojedinog prosječnog trajanja prijenosa, poredak mjerenja po prosječnom trajanju i standardnoj devijaciji te druge slične obrade ukupnih rezultata.Nakon obrade pojedini rezultati se prosljeđuju posebnim funkcijama za prikaz ukupnih podataka. Funkcije za tablični prikaz podataka podatke ispisuju u obliku HTML tablica. Funkcije za grafički prikaz podataka vrše potrebno kodiranje podataka u URL identifikatore koji pokazuju na servis za prikaz grafova tvrtke Google - Google Chart Server. Dobiveni URL identifikatori predaju se Web pregledniku koji dohvaća slike od Google Chart Server poslužitelja te ih zatim prikazuje. Grafički prikaz podržava razne načine prikaza rezultata. Korisniku je pružen na odabir stupčasti ili linijski prikaz (Slika 4.5) sa raznim grafičkim stilovima. Najtočniji prikaz je stupčasti prikaz. Kod linijskog prikaza valja imati na umu da su mjerenja izvršena samo na označenim točkama dok su linije koje ih spajaju samo linearna interpolacija koja u području eksponencijalnog rasta vremena trajanja može znatno odudarati od stvarnih vrijednosti koje bi bile izmjerene u tom mjerenju. Ukoliko je broj mjerenja dovoljno velik te su mjerenja relativno blizu, linearna procjena daje dobar uvid u kretanje vremena.



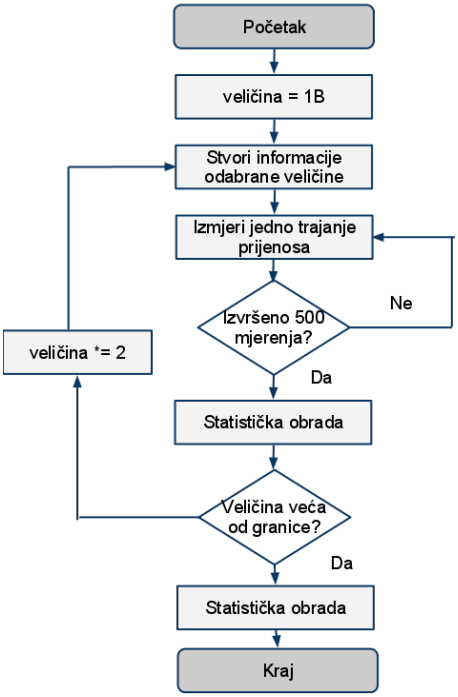
(a) Stupčasti **(b)** Linijski

Slika 4.5: Stupčasti i linijski prikaz

4.6. Upravljanje tokom izvršavanja mjerenja

Tok mjerenja prikazan je na slici 4.6. Mjerenje se sastoji od dvije velike logičke cjeline: uzastopnog i istodobnog scenarija. U uzastopnom scenariju svaki prijenos informacije se blokira sve dok se za prošlu ne primi potvrda o primitku. U istodobnom scenariju ne čeka se potvrda o primitku već se poruke šalju jedna za drugom. Slanje potvrda o primitku poruka u toku istodobnog mjerenja može se izmješati sa slanjem izvornih poruka, ovisno o mehanizmu te nasljedno asinkronom modelu izvršavanja više različitih konteksta u Web pregledniku. Tok izvršavanja uzastopnog scenarija prikazan je na slici 4.7. Uzastopni scenarij sastoji se od niza mjerenja. Svako mjerenje sastoji se od 500 ponavljanja osnovnog mjerenja prikazanog na 3.4. Svako novo veliko mjerenje od prošlog se razlikuje po veličini jedne poruke. Veličina informacija u poruci raste eksponencijalno s faktorom 2 a počine na 1B. Uz informacije svaka poruka također prenosi i 4B identifikacijske oznake. Nakon svakog pojedinog niza uzastopnih mjerenja izvedenih za određenu količinu informacija vrši se statistička obrada izvršenih mjerenja. Po završetku izvedbe svih uzastopnih mjerenja za sve željene veličine vrši se zbirna statistička obrada svih mjerenja. Istodobni scenarij započinje nakon zbirne statističke obrade uzastopnog scenarija. Kao i kod uzastopnog scenarija i u istodobnom scenariju se vrši niz mjerenja, svaki put s različitom veličinom informacija u poruci. Veličina informacija raste eksponencijalno s faktorom 2 a počinje na 1B. Tok istodobnog scenarija prikazan je na 4.8. Za razliku od uzastopnog scenarija, posšiljatelj ne čeka potvrdu primitka svake poruke već ih šalje najvećom brzinom koju mu mehanizam i Web preglednik dopuštaju. Slanje i primanje poruka u istodobnom scenariju može se odvijati istodobno, te je moguće da će se poruke primati drugačijim redosljedom od kojeg su poslane. Radni okvir stoga čeka sve dok ne primi potvrdu o primitku za svaku poslanu poruku. Kad primi sve potvrde, vrši statističku obradu izvršenog mjerenja te povećava

Slika 4.6: Opći tok mjerenja radnih svojstava jednog mehanizma. Uzastopni i istodobni scenarij su složeni koraci prikazani na 4.7 i 4.8.



Slika 4.7: Tok uzastopnog scenarija.

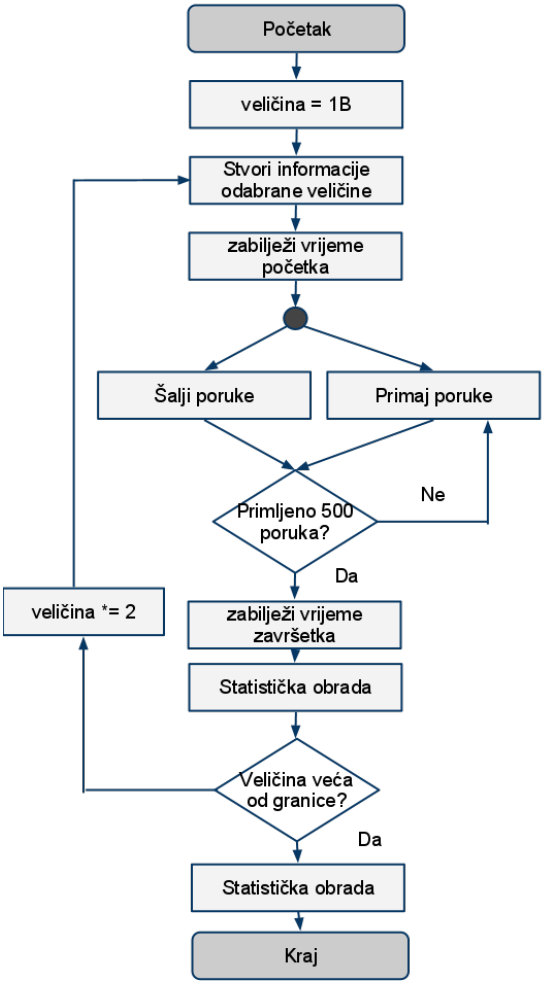
veličinu poruke.

4.7. Primjeri korištenja radnog okvira

U ovom poglavlju opisani su primjeri korištenja radnog okvira sa korisničke strane. Prikazani su komunikacijski mehanizmi postMessage i pmrpc te izravno dijeljenje varijabli unutar istog konteksta. Za svaki primjer prikazan je izvršni kod potreban za stvaranje sučelja prema radnom okviru te kratki osvrt na posebnosti svakog sučelja.

4.7.1. Izravno dijeljene varijable

Izravno dijeljene varijable su rubni slučaj komunikacije. Trajanje pridruživanja vrijednosti varijabli je kraće od najfinije mjerne rezolucije radnog okvira, stoga rezultati za izravno dijeljene varijable nisu značajni. Način izvedbe sučelja s radnim okvirom međutim dobro prikazuje osnovnu zamisao radnog okvira te je dobra polazna točka za druge mehanizme. Izvorni kod sučelja prikazan je na 4.9.



Slika 4.8: Tok istodobnog scenarija.

```
 1 <html>
 2   <head>
 3     <script type="text/javascript" src="cccmpbm.js"/>
 4     <script tpye="text/javascript">
 5         cccmpbm.send = function (id, msg) {
 6             cccmpbm.recv(id, msg);
 7         }
 8
 9         cccmpbm.recv = function (id, msg){
10             cccmpbm.recvRet(-id, msg);
11         }
12     </script>
13   </head>
14   <body onLoad="cccmpbm.runTests('results');">
15     <div id="results">
16     </div>
17   </body>
18 </html>
```

Slika 4.9: Implementacija sučelja za izravno dijeljene varijable. U liniji 5 proširena je send funkcija sučelja. U liniji 7 proširena je recv funkcija sučelja. U liniji 6 funkcija send direktno poziva funkciju recv, te se varijable id i msg direktno dijele između pošiljatelja i primatelja.

4.7.2. postMessage

Mehanizam postMessage je ugrađen u Web preglednike koji sadrže izvedbe HTML5 skupa standarda. Mehanizam postMessage izložen je kao ugrađena JavaScript funkcija i događaj Web preglednika. Detaljan opis mehanizma dan je u poglavlju 2.4.1. Odsječak programskog koda 4.10 prikazuje programski kod za oba konteksta korištena u mjerenju. Oba konteksta moraju implementirati obje funkcije sučelja. Kontekst A funkcijom send šalje izvornu poruku, a funkcijom recv prima potvrde o primitku. Kontekst B funkcijom send šalje potvrde o primitku, a funkcijom recv prima izvorne poruke.

4.7.3. pmrpc

Mehanizam pmrpc je zasnovan na udaljenom pozivu procedura. On je nadgradnja nad mehanizmom postMessage te izlaže drugačiji model komunikacije te dodatna svojstva poput pouzdanosti, imenovanja i otkrivanja. Detaljan opis pmrpc mehanizam nalazi se u poglavlju 2.4.2. Odsječak programskog koda 4.12 prikazuje implementaciju sučelja prema radnom okviru za oba konteksta. Kod modela zasnovanog na pozivu udaljenih procedura izvorni kontekst A mora implementirati samo funkciju send. Potvrda o primitku ne šalje se iz konteksta B ponovnim pozivom funkcije send budući da se povratna vrijednost prilikom poziva udaljenih procedura prenosi preko komunikacijskog kanala sve do pozivatelja. Zbog toga kontekst B implementira samo funkciju recv čija povratna vrijednost je potvrda o primitku.

```
 1 <html>
 2   <head>
 3     <script type="text/javascript" src="cccmpbm.js"/>
 4     <script type="text/javascript">
 5         cccmpbm.send = function (id, msg) {
 6             window.frames[0].postMessage(
 7                 JSON.stringify({'id': id, 'msg': msg}),
 8                 "*");
 9         }
10
11         function recv(e){
12             data = JSON.parse(e.data);
13             cccmpbm.recvRet(data['id'], data['msg']);
14         }
15
16         window.addEventListener("message", recv, false);
17     </script>
18   </head>
19   <body onLoad="cccmpbm.runTests('results');">
20     <iframe src="postMessageServer.html"></iframe>
21     <div id="results">
22     </div>
23   </body>
24 </html>
```

Slika 4.10: Implementacija izvornog konteksta za mehanizam postMessage. Izvorni kontekst za mehanizam postMessage u linijama 5 - 9 proširuje send funkciju sučelja. Pozivom postMessage funkcije objekta koji opisuje odredišni kontekst šalje se poruka. U liniji 16 vrši se pretplata na onMessage događaj kojim će se primiti potvrda o primitku od primatelja.

```
 1 <html>
 2   <head>
 3     <script type="text/javascript" src="cccmpbm.js"/>
 4     <script type="text/javascript">
 5         cccmpbm.send = function (id, msg) {
 6             window.parent.postMessage(
 7                 JSON.stringify({'id': -id, 'msg': msg}),
 8                 "*");
 9         }
10
11         function recv(e){
12             data = JSON.parse(e.data);
13             cccmpbm.recvRet(data['id'], data['msg']);
14         }
15         window.addEventListener("message", recv, false);
16     </script>
17   </head>
18   <body>
19     <div>
20         postMessage server here!
21     </div>
22   </body>
23 </html>
```

Slika 4.11: Implementacija odredišnog konteksta za mehanizam postMessage. Implementacija sučelja odredišta slična je implementaciji izvorišta prikazanoj u 4.10

```
 1 <html>
 2   <head>
 3     <script type="text/javascript" src="cccmpbm.js"/>
 4     <script type="text/javascript" src="pmrpc.js"/>
 5     <script type="text/javascript">
 6         cccmpbm.send = function (id, msg) {
 7             pmrpc.call({
 8                 destination: window.frames[0],
 9                 publicProcedureName: "recv",
10                 params: [id, msg],
11                 onSuccess: function(result) {
12                     cccmpbm.recvRPC(
13                         result.returnValue[0],
14                         result.returnValue[1]);
15                 },
16                 onError: function(statusObj) {
17                     alert("broken");
18                 }
19             });
20         }
21     </script>
22   </head>
23   <body onLoad="cccmpbm.runTests('results');">
24     <iframe src="pmrpcServer.html"></iframe>
25     <div id="results">
26     </div>
27   </body>
28 </html>
```

Slika 4.12: Implementacija izvornog konteksta za mehanizam pmrpc. U liniji 7 započinje poziv udaljene procedure. Parametri poziva predaju se kao riječnik čije se vrijednosti pune u linijama 8 - 18. Parametri su redom: odredište, naziv procedure, parametri koji će se predati udaljenoj proceduri, povratna funkcija koja će se pozvati u slučaju uspješnog izvršavanja poziva te povratna funkcija koja će se pozvati u slučaju pogreške.

```
 1 <html>
 2   <head>
 3     <script type="text/javascript" src="cccmpbm.js"/>
 4     <script type="text/javascript" src="pmrpc.js"/>
 5     <script type="text/javascript">
 6         pmrpc.register({
 7             publicProcedureName: "recv",
 8             procedure : function(id, msg) {
 9                 var time = (new Date()).getTime();
10                 return [id, time];
11             }
12         });
13     </script>
14   </head>
15   <body>
16     <div>
17         pmrpc server here!
18     </div>
19   </body>
20 </html>
```

Slika 4.13: Implementacija odredišnog konteksta za mehanizam pmrpc. Pozivom naredbe register registrira se udaljena procedura spremna za pozivanje.

5. Rezultati

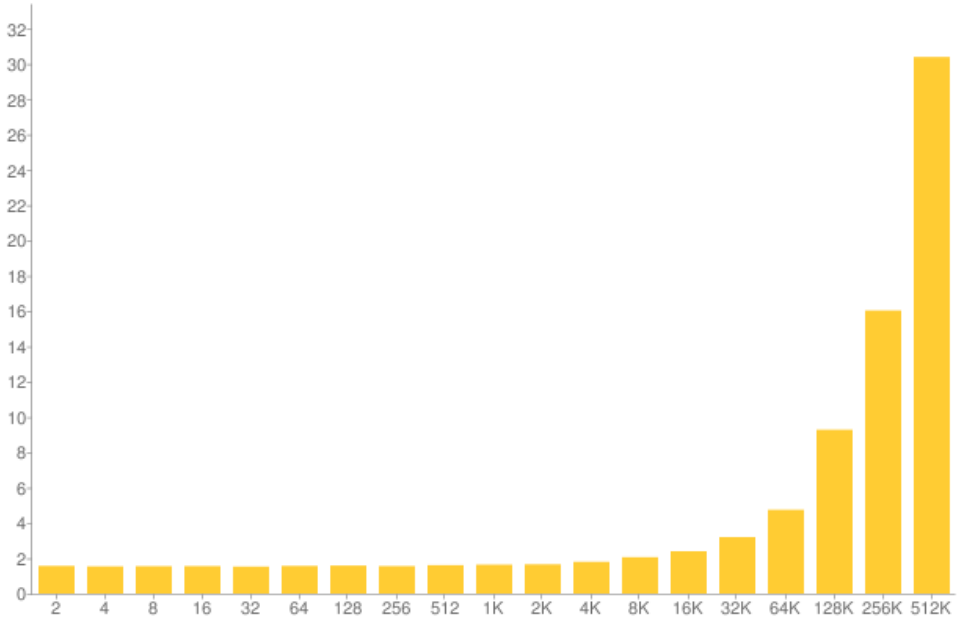
U ovom poglavlju predstavljeni su rezultati dobiveni mjrenjem radnih svojstava komunikacijskih mehanizama. Za svaki mehanizam iznesena su pojedinačna radna svojstva te kratka analiza rezultata s ciljem izdvajanja mogućih uzroka. Usporedni rezultati svih mehanizama također su prikazani i analizirani. Kod složenih mehanizama koji koriste jednostavne mehanizme za prijenos podataka dana je usporedba s odgovarajućim jednostavnim mehanizmima.

5.1. Mehanizam postMessage

Mehanizam postMessage je jednostavan mehanizam ugrađen u Web preglednik, te se zbog toga koriti kao transportni mehanizam za složenije mehanizme. Mjerna svojstva mehanizma postMessage djeluju dakle ograničavajuće na radna svojstva složenijih mehanizama te su stoga od posebne važnosti. Graf 5.1 prikazuje srednje vrijeme trajanja slanja poruke. Na osi apcise unesene su veličine prenešenih poruka, počevši od 2B pa do 512KB. Iz grafa se vidi da vrijeme prijenosa raste linearno s veličinom poruke.

Linearni rast zadržava se i na porukama reda veličine nekoliko megabajta. Graf 5.2 prikazuje sredjnje vrijeme trajanja slanja poruke za veličine do 8MB. Kod veličina većih od nekoliko desetaka megabajta, vrijeme prenosa slanja nije moguće pouzdano izmjeriti budući da, u ovisnosti o računalu i operativnom sustavu, Web preglednici ulaze u područje nepouzdanosti te sigurnosni sustavi Web preglednika detektiraju nepouzdano ponašanje te zaustave izvršavanje programskog koda.

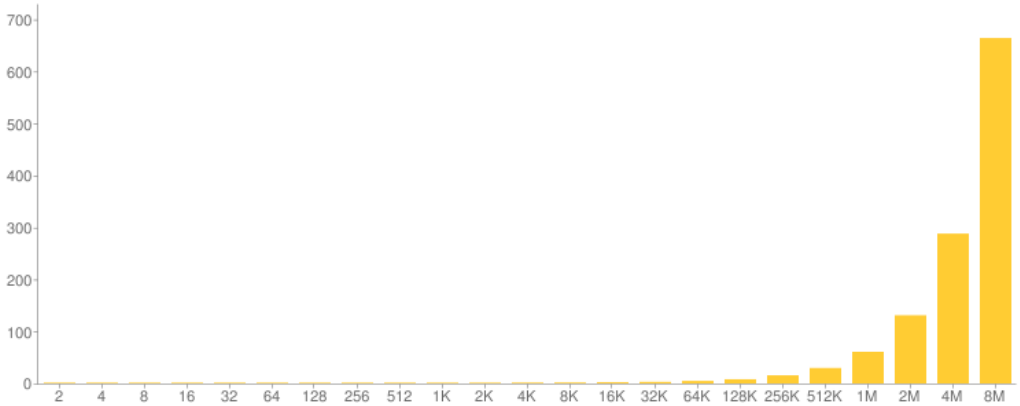
Prijenos jedne poruke korištenjem mehanizma postMessage dakle traje proporcionalno veličine poruke. Za veliku većinu trenutnih potreba ograničenje reda veličine na 10MB je dovoljno te je stoga za veliku većinu potreba trajanje prijenosa jedne poruke manje od 1s. Vrlo veliki broj Web aplikacija razmjenjuje poruke manje od 1MB, čije prosječno vrijeme trajanja prenosa je reda veličine 50ms. Web aplikacije koje ne šalju poruke veće od 1KB imaju u praktičnom smislu trenutan prijenos, reda veličine 1ms. Istodobnim scenarijem testiranja pokazuje se ponašanje mehanizma u slučaju



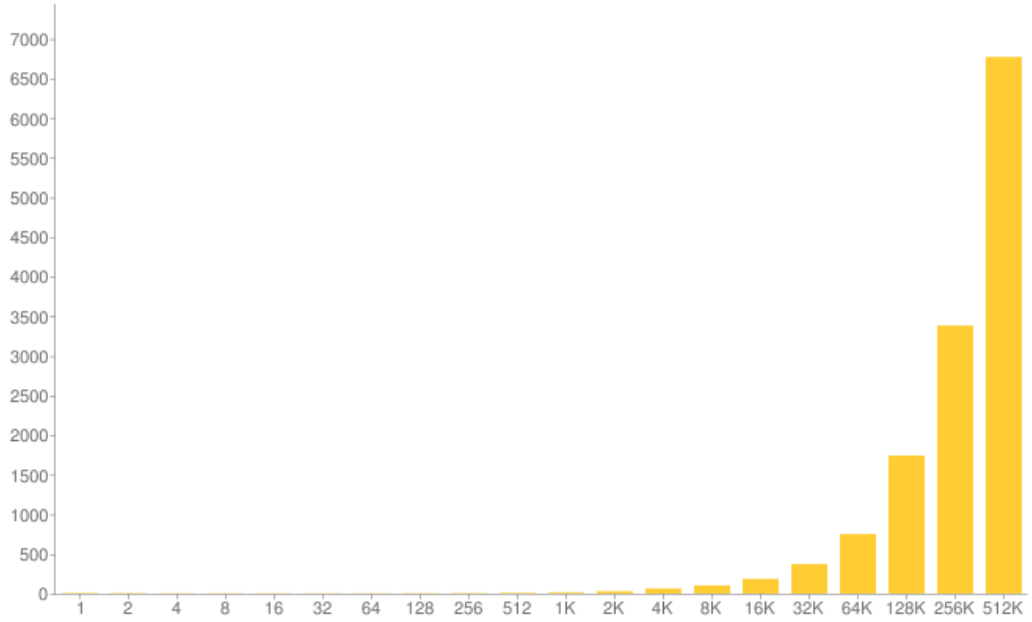
Slika 5.1: Prosječno vrijeme trajanja slanja poruke za mehanizam postMessage - do 512KB. Os apcisa označava količinu prenesenih informacija, os ordinata prosječno trajanje prijenosa.

više istodobnih poruka. Graf 5.3 prikazuje ukupno vrijeme potrebno za prenos 100 poruka veličine 1B do 512KB. Eksponencijalni rast veličine poruke odgovara eksponencijalnom rastu vremena prijenosa.

Usporedba grafova 5.3 i 5.1 zanimljiva je zbog toga što pokazuje odnos istodobne i uzastopne komunikacije. Graf 5.4 prikazuje usporedbu istodobne i uzastopne komunikacije. Pri veličinama poruka manjima od 32KB istodobnaa komunikacija je nekoliko redova veličine brža od uzastopne komunikacije. Brzina porasta trajanja istodobne komunikacije je međutim veća od uzastopne komunikacije. Pri veličini od 32KB isto-



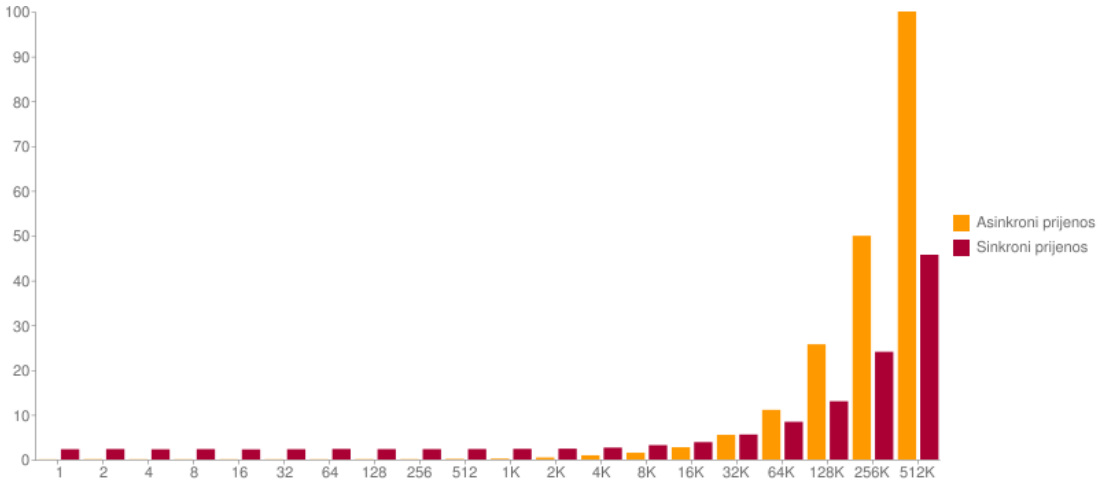
Slika 5.2: Prosječno vrijeme trajanja slanja poruke za mehanizam postMessage - do 8MB. Os apcisa označava količinu prenesenih informacija, os ordinata prosječno trajanje prijenosa.



Slika 5.3: Prosječno ukupno vrijeme trajanja istodobnog prijenosa 100 poruka veličine do 512BK. Os apcisa označava količinu prenesenih informacija, os ordinata prosječno ukupno trajanje.

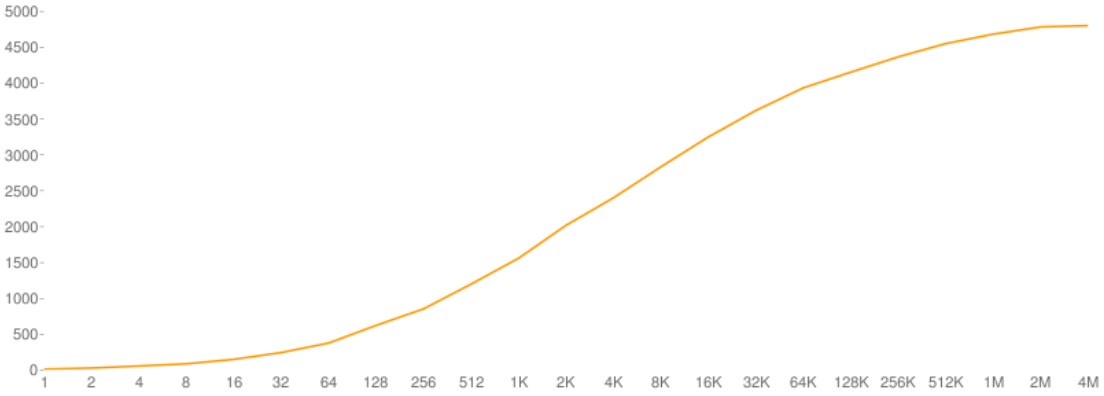
dobni prijenos 100 poruka duljeg je trajanja od uzastopnog prijenosa 100 poruka. Sa porastom veličine poruke razlika postaje sve izraženija. Pri 512KB istodobna komunikacija je 50% sporija od uzastopne.

Usporavanje istodobne komunikacije kod većih poruka rezultat je ograničenih memorijskih resursa Web preglednika. Za spremanje 100 poruka veličine 512KB Web preglednik mora odvoiti približno 50MB radne memorije, dok kod uzastopnog prijenosa u svakom trenutku treba samo 512KB. Kod preglednika koji ne vrše memorijsku optimizaciju ukupna vršnja potrošnja radne memorije može dostići i 100MB u trenutku stvaranja odredišnih poruka. Iako red veličine manje od ukupne raspoložive memorije računala, koja se mjeri u GB, procesorsko vrijeme potrebno za upravljanje s desecima ili stotinama MB memorije znatno utječe na radna svojstva istodobne komunikacije. Graf 5.5 prikazuje ukupnu količinu prenesenih podataka podijeljenu s ukupnim trajanjem prijenosa. Dobiveni rezultat je iskorištena širina prijenosa prikazana žutom linijom. Na grafu je jasno vidljiv porast korištene širine prijenosa sve dok se ne dostigne najveća moguća širina prijenosa otprilike pri veličini poruke od 1MB. Nakon toga slijedi područje potpunog iskorištavanja moguće širine prijenosa. Kod veličine poruke od 16MB Web preglednik treba 1.6GB u optimiziranom ili 3.2GB u neoptimiziranom slučaju što je više nego što mu operativni sustav može pružiti te dolazi do naglog pada radnih svojstava zbog korištenja priručne memorije na tvrdom disku. Zbog naglog



Slika 5.4: Usporedba trajanja prenosa 100 poruka u uzastopnom i istodobnom scenariju. Os apcisa označava količinu prenesenih informacija a os ordinata prosječno ukupno trajanje.

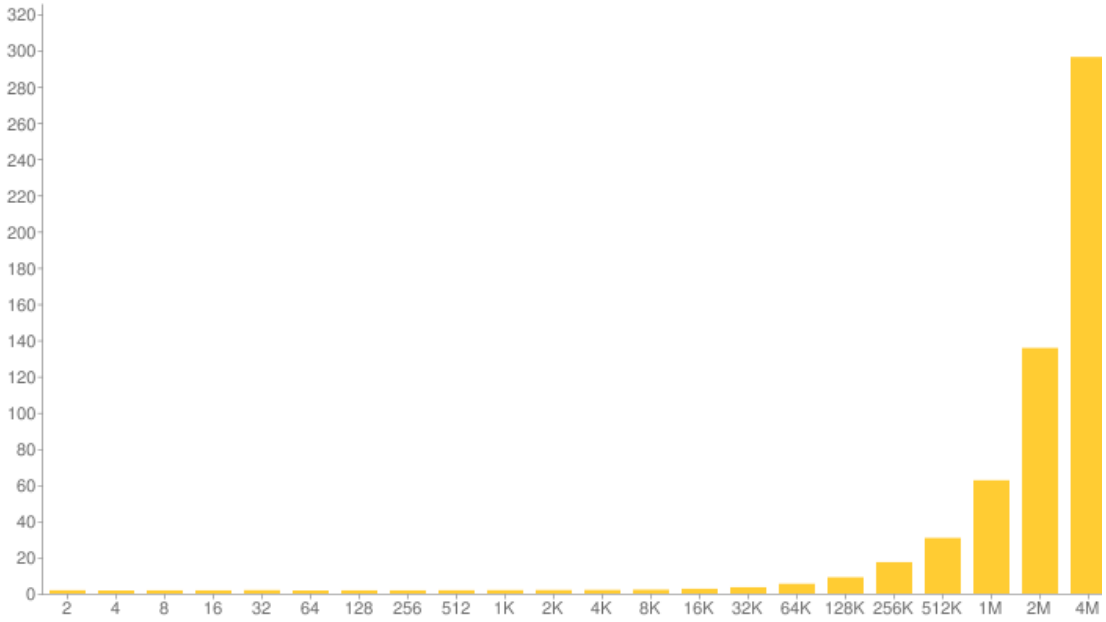
pada radnih svojstava mehanizam postaje neupotrebljiv, do razine na kojoj je i mjerenje vremena trajanja nepraktično zbog vrlo velikih rezultata koji se izražavaju u satima.



Slika 5.5: Širina prijenosa mehanizma postMessage. Na osi apcisa navedena je veličina poruke, a na osi ordinata širina prijenosa u KB/s.

5.2. Mehanizam pmrpc

Mehanizam pmrpc koristi mehanizam postMessage kao transportni mehanizam te se stoga očekuju slični rezultati. Uzastopni prijenos do 8MB prikazan na slici 5.6 potvrđuje takve slutnje. Na slici se vidi linearan porast vremena prijenosa u odnosu na količinu informacija.



Slika 5.6: Prosječno vrijeme trajanja slanja poruke za mehanizam pmrpc - do 8MB. Os apcisa označava količinu prenesenih informacija, os ordinata prosječno trajanje prijenosa.

Rezultati istodobnog scenarija prikazani su na slici 5.7. Na slici je vidljivo da mehanizam pmrpc može koristiti istodobni prijenos informacija za poboljšanje radnih svojstava za male količine informacija. Kod određene količine informacija, točan broj varira u ovisnosti od računala i operativnog sustava, mehanizam pmrpc koristi svu raspoloživu širinu prijenosa te za veće količine informacija radna svojstva stagniraju ili postaju lošija.

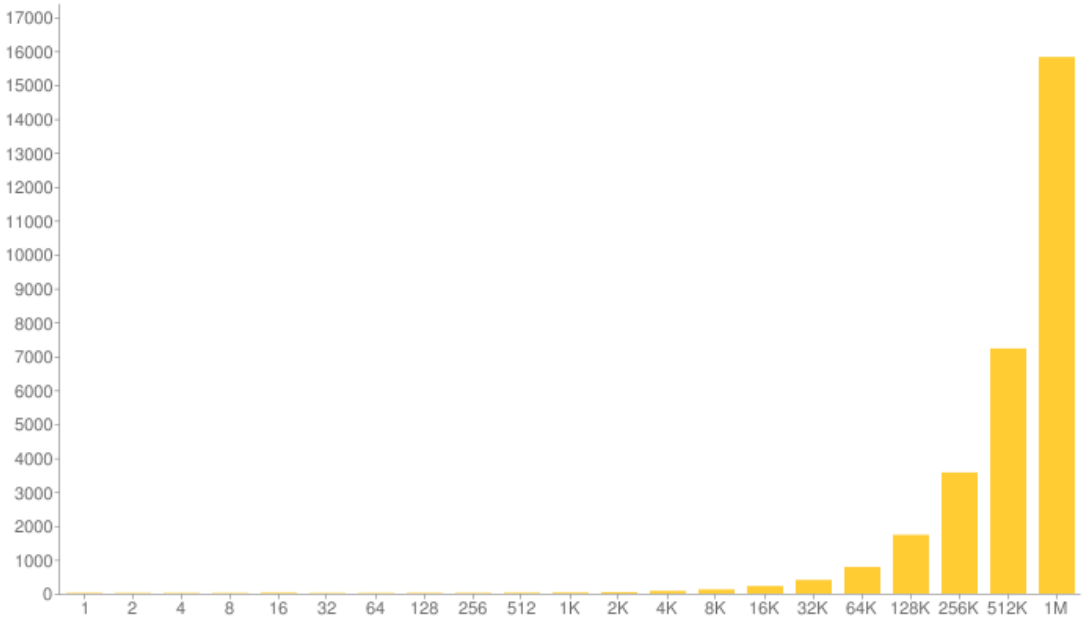
5.3. Usporedba mehanizama

Radna svojstva mehanizma postMessage i pmrpc uspoređena su u tablici 5.1. Prvi

Tablica 5.1: Usporedba radnih svojstava mehanizma postMessage i pmrpc.

Radno svojstvo	postMessage	pmrpc	Apsulutna razlika	Relativna razlika
Uzastopni - 512KB	30ms	32ms	2ms	6.6%
Uzastopni - 8MB	648ms	675ms	27ms	4.1%
Istodobni - 512KB	6570ms	7213ms	643ms	9.7%
Istodobni - točka zasićenja	2MB	1MB	1MB	100%

redak tablice uspoređuje uzastopni scenarij, 512KB informacija. Mehanizam pmrpc je približno 5% sporiji od mehanizma postMessage. U apsulutnim terminima razlika



Slika 5.7: Prosječno ukupno vrijeme trajanja istodobnog prijenosa 100 poruka veličine do 1MB. Os apcisa označava količinu prenesenih informacija, os ordinata prosječno ukupno trajanje.

je zanemarivih 2ms. Drugi redak tablice uspoređuje uzastopni scenarij, 8MB informacija. Apsulutna razlika je i dalje zanemariva, 27ms. Relativna razlika je i dalje približno 5%. Treći redak uspoređuje istodobni scenarij, s 512KB informacija. Mehanizam pmrpc je ovdje 10% sporiji od mehanizma postMessage. U apsulutnim terminima pmrpc je više od pola sekunde sporiji. U primjeni koja korisniku prikazuje rezultate u stvarnom vremenu, pola sekunde bi moglo biti značajno usporenje. Četvrti redak uspoređuje količinu informacija pri kojoj mehanizam koristi punu širinu prijenosa. Mehanizam pmrpc duplo brže iskorištava širinu prijenosa, zbog toga što lošije gospodari s memorijskom prostorom.

Za aplikacije koje prenose manje količine podataka, radna svojstva mehanizma pmrpc nisu osjetno lošija od radnih svojstava mehanizma postMessage. Dodatne mogućnosti dobivene korištenjem opsežnijeg mehanizma u tom slučaju su opravdane.

6. Zaključak

Web preglednici su već postali platforma za razvoj aplikacija ravnopravna sa operativnim sustavima. S dolaskom operativnih sustava zasnovanih na Web preglednicima, poput Chrome OS-a, te mobilnih uređaja koji omogućavaju djelomičnu integraciju Web aplikacija na svoje radne površine, poput Android i iPhone uređaja, granice i razlike između Web aplikacija i izvornih aplikacija biti će sve zamućenije. U skladu s tim znanstveno istraživanje osnovnih infrastrukturnih elemenata na kojima počivaju Web aplikacije mora se postaviti na istu razinu koju trenutno imaju izvorne aplikacije operativnih sustava. U prošlosti se većina istraživanja vezanih uz svojstva Web aplikacija bavila problemima sigurnosti, s rijetkim udjelom funkcionalnih svojstava. Istraživanje radnih svojstava Web aplikacija vrlo je rijetko te većinom ograničeno na brzinu izvršavanja JavaScript skriptnog koda. Komunikacija unutar Web preglednika je obrađena tek u nekoliko radova.

Komunikacija unutar Web preglednika ima snažne paralele s međuprocesnom komunikacijom u klasičnim operativnim sustavima. Moderni Web preglednici skriptni kod u jeziku JavaScript izvršavaju paralelno u svakom kontekstu na način koji zrcali procesno izvršavanje u operativnom sustavu, a Web radnici prvi put uvedeni u HTML5 skupu standarda odgovarajući su element klasičnih dretvi. Komunikacija među različitim kontekstima ili Web radnicima nužna je za stvaranje modularnih Web aplikacija te znatno doprinosi mogućnostima razvoja Web aplikacija. S obzirom na veliku heterogenost različitih komunikacijskih mehanizama potrebna je sistematizacija svojstva te ujednačena ocjena radnih svojstva svakog mehanizma. Kako bi se uspješno nosila s vrlo dinamičnim ekosustavom Web preglednika ocjena radnih svojstva mora biti u potpunosti automatizirana.

U ovom radu predstavljen je radni okvir koji automatizirano mjeri, ocjenjuje i uspoređuje radna svojstva raznih komunikacijskih mehanizama. Rezultati dobiveni radnim okvirom su prilagođeni za daljnju analizu i prikazani na način koji omogućuje inžinjerima i razvijateljima Web aplikacija brzu odluku o mehanizmu prikladnom za njihove potrebe. Detaljnija analiza rezultata prikazuje neke uzorke prisutne u više mehanizama

iz kojih je moguće zaključiti neka svojstva koja bi trebao imati idealni komunikacijskih mehanizam. Idealni mehanizam trebao bi imati linearni porast odziva u odnosu na količinu informacija, te mogućnost iskorištavanja istodobnog prijenosa u svrhu poboljšanja radnih svojstava. Upravljanje memorijskim prostorm jedan je od većih čimbenika u dobivenim rezultatima te bi se pri izradi mehanizma posebnu pažnju trebalo obratiti na smještaj informacija u memorijskom prostoru.

Otvorena pitanja iznesena u ovom radu koja je potrebno obraditi u budućnosti su utjecaj sigurnosnih mogućnosti mehanizma na radna svojstva, mogućnost poboljšanja radnih svojstava boljim iskorištavanjem istodobnog prijenosa informacija, izvedba radnog okvira koji mjeri točnost u odnosu na referentnu vremensku vrijednost, mjerenje memorijskih radnih svojstava poput potrošnje, razmještaja i brzine pristupa te mjerenje složenosti upotrebe mehanizma kroz mjere složenosti programskog koda. Osim poboljšanja mehanizama i radnog okvira također je potrebno radni okvir primjeniti na što većem skupu mehanizama kako bi se pokazalo općenito stanje i trendovi ekosustava mehanizama.

LITERATURA

Jeremy Allaire. Macromedia flash mx—a next-generation rich client. Technical report, Macromedia, 2002. URL http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf.

Adam Barth, Collin Jackson, i John C. Mitchell. Securing frame communication in browsers. *Commun. ACM*, 52:83–91, June 2009. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/1516046.1516066. URL http://doi.acm.org/10.1145/1516046.1516066.

Ori Berger. Windows timer precision?, November 2003. URL http://discuss.fogcreek.com/joelonsoftware/default.asp?cmd=show&ixPost=85520.

Tim Berners-Lee i Robert Cailliau. WorldWideWeb: Proposal for a HyperText project. Technical report, CERN, 1990. URL http://www.w3.org/Proposal.

Piero Fraternali, Gustavo Rossi, i Fernando Sánchez-Figueroa. Rich internet applications. *IEEE Internet Computing*, 14(3):9–12, 2010. URL http://doi.ieeecomputersociety.org/10.1109/MIC.2010.76.

Dieter Gollmann. Computer security. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(5):544–554, 2010. ISSN 1939-0068. doi: 10.1002/wics.106. URL http://dx.doi.org/10.1002/wics.106.

Ian Hickson. Web workers, June 2011. URL http://dev.w3.org/html5/postmsg/.

Marko Ivanković. Can javascipt gettime() be used for measuring performance of javascript code?, March 2011. URL http://mivankovic.blogspot.com/2011/03/research-results-can-javascipt-gettime.html.

Charles Reis. Improving the security and robustness of modern web browsers abstract. Technical report, Kolovoz 14 2008. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.2006;http://www.cs.washington.edu/homes/creis/publications/generals-report.pdf.

John Resig. Accuracy of javascript time, November 2008. URL http://ejohn.org/blog/accuracy-of-javascript-time/.

Kapil Singh, Alexander Moshchuk, Helen J. Wang, i Wenke Lee. On the incoherencies in web browser access control policies. stranice 463–478, 2010a. URL http://doi.ieeecomputersociety.org/10.1109/SP.2010.35.

Kapil Singh, Alexander Moshchuk, Helen J. Wang, i Wenke Lee. On the incoherencies in web browser access control policies. *Security and Privacy, IEEE Symposium on*, 0:463–478, 2010b. ISSN 1081-6011. doi: http://doi.ieeecomputersociety.org/10.1109/SP.2010.35.

Siniša Srbljić, Dean Škvorc, i Daniel Skrobo. Widget-oriented consumer programming. *AUTOMATIKA: Journal for Control, Measurement, Electronics, Computing and Communication*, 50(3-4):252–264, 2009.

Alex Wright. Ready for a web OS? *Commun. ACM*, 52(12):16–17, 2009. URL http://doi.acm.org/10.1145/1610252.1610260.

Ivan Žužak, Marko Ivanković, i Ivan Budiselić. Cross-context web browser communication with unified communication models and context types. U *MIPRO CTS - Computers in Technical Systems*, 2011.

Radni okvir za mjerenje radnih svojstava za komunikaciju između konteksta Web preglednika

Sažetak

Komunikacija u Web pregledniku dobiva na važnosti sa porastom popularnosti Weba kao programske platforme. S dolaskom operativnih sustava baziranih na Web preglednicima, poput Chrome OS-a, komunikacija u Web preglednicima postaje ekvivalentna međuprocesnoj komunikaciji u klasičnim operativnim sustavima. Klasični operativni sustavi predmet su brojnih istraživanja te su vrlo dobro poznata funkcijska, sigurnosna i radna svojstva međuprocesne komunikacije. Za razliku od njih komunikacijski mehanizmi Web preglednika su dobro istraženi samo po svojim sigurnosnim svojstvima dok su funkcionalna pokrivena površno, a radna svojstva nisu pokrivena uopće. U ovom radu predstavljen je radni okvir koji olakšava mjerenje radnih svojstava komunikacijskih mehanizama te njihovu usporedbu. Prikazan je primjer upotrebe radnog okvira te su dani prvi rezultati za neke poznatije mehanizme.

Ključne riječi: Web preglednici, međukontekstna komunikacija, radna svojstva, radni okvir

Software framework for performance benchmarks of cross-context communication systems for Web browsers

Abstract

Web browsers are becoming more and more important as the idea of Web as an application platform gains track. In 2011 we saw the release of first fully Web browser based operating system - Chrome OS. In such operating systems the role of traditional interprocess comunication is replaced by cross-context communication systems contained within the browser. While traditional interprocess communication mechanisms were the topic of much research from security, functionality and performance standpoints, research on cross-context communication dealt primarily with security issues. A handfull of papers explored functional aspects and none explored performance issues. In this thesis a framework is presented that enables simple consistent performance benchmarking of different cross-context communication systems. Several prominent systems are evaluated and the results are analysed.

Keywords: Web browsers, cross-context communication, performance, benchmark, software framework