UNIVERSITY OF NOVI SAD

FACULTY OF TECHNICAL SCIENCES
NOVI SAD

Miloš Simić

Micro clouds and edge computing as a service

- Ph. D. Thesis -

Supervisor
Goran Sladić, PhD, associate professor

Novi Sad, 2021.

Miloš Simić: *Micro clouds and edge computing as a service*

SERBIAN TITLE: Micro clouds and edge computing as a service

SUPERVISOR:
Goran Sladić, PhD, associate professor

LOCATION:
Novi Sad, Serbia

DATE:
September 2021

UNIVERZITET U NOVOM SADU • **FAKULTET TEHNIČKIH NAUKA**
21000 NOVI SAD, Trg Dositeja Obradoviće 6

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Redni broj, **RBR**:	
Identifikacioni broj, **IBR**:	
Tip dokumentacije, **TD**:	Monografska dokumentacija
Tip zapisa, **TZ**:	Tekstualni štampani materijal
Vrsta rada, **VR**:	Doktorska disertacija
Autor, **AU**:	Miloš Simić
Mentor, **MH**:	dr Goran Sladić, vanredni profesor
Naslov rada, **NR**:	Micro clouds and edge computing as a service
Jezik publikacije, **JP**:	engleski
Jezik izvoda, **JI**:	srpski
Zemlja publikacije, **ZP**:	Srbija
Uže geografsko područije, **UGP**:	Vojvodina
Godina, **GO**:	2021
Izdavač, **IZ**:	Fakultet tehničkih nauka
Mesto i adresa, **MA**:	Trg Dositeja Obradovića 6, 21000 Novi Sad
Fizički opis rada, **FO**: (poglavlja/strana /citata/tabela/slika/grafika/priloga)	6/159/154/13/13/0/0
Naučna oblast, **NO**:	Elektrotehničko i računarsko inženjerstvo
Naučna disciplina, **ND**:	Distribuirani sistemi
Predmetna odrednica/Ključne reči, **PO**:	distributed systems, cloud computing, microservices, software as a service, edge computing, micro clouds
UDK	
Čuva se, **ČU**:	Biblioteka Fakulteta tehničkih nauka, Trg Dositeraj Obradovića 6, 21000 Novi Sad
Važna napomena, **VN**:	

UNIVERZITET U NOVOM SADU • **FAKULTET TEHNIČKIH NAUKA**
21000 NOVI SAD, Trg Dositeja Obradoviće 6

KLJUČNA DOKUMENTACIJSKA INFORMACIJA

Izvod, **IZ**:		U sklopu disertacije izvršeno je istraživanje u oblasti razvoja bezbednog softvera. Razvijene su dve metode koje zajedno omogućuju integraciju bezbednosne analize dizajna softvera u proces agilnog razvoja. Prvi metod predstavlja radni okvir za konstruisanje radionica čija svrha je obuka inženjera softvera kako da sprovode bezbednosnu analizu dizajna. Drugi metod je proces koji proširuje metod bezbednosne analize dizajna kako bi podržao bolju integraciju spram potreba organizacije. Prvi metod je evaluiran kroz kontrolisan eksperiment, dok je drugi metod evaluiran upotrebom komparativne analize i analize studija slučaja, gde je proces implementiran u kontekstu dve organizacije koje se bave razvojem softvera.	
Datum prihvatanja teme, **DP**:			
Datum odbrane, **DO**:			
Članovi komisije, **KO**:	Predsednik:	dr Branko Milosavljević, redovni profesor, FTN, Novi Sad	
	Član:	dr Silvia Gilezan, redovni profesor, FTN, Novi Sad	
	Član:	dr Gordana Milosavljević, vanredni profesor, FTN, Novi Sad	Potpis mentora
	Član:	dr Žarko Stanisavljević, docent, ETF, Beograd	
	Član, mentor:	dr Goran Sladić, vanredni profesor, FTN, Novi Sad	

UNIVERSITY OF NOVI SAD • **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:	
Identification number, **INO**:	
Document type, **DT**:	Monograph documentation
Type of record, **TR**:	Textual printed material
Contents code, **CC**:	Ph.D. thesis
Author, **AU**:	Miloš Simić
Mentor, **MN**:	Goran Sladić, Ph.D., Associate Professor
Title, **TI**:	Micro clouds and edge computing as a service
Language of text, **LT**:	English
Language of abstract, **LA**:	Serbian
Country of publication, **CP**:	Serbia
Locality of publication, **LP**:	Vojvodina
Publication year, **PY**:	2021
Publisher, **PB**:	Faculty of Technical Sciences
Publication place, **PP**:	Trg Dositeja Obradovića 6, 21000 Novi Sad
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/	6/159/154/13/13/0/0
Scientific field, **SF**:	Electrical engineering and computing
Scientific discipline, **SD**:	Distributed systems
Subject/Key words, **S/KW**:	distributed systems, cloud computing, microservices, software as a service, edge computing, micro clouds
UC	
Holding data, **HD**:	Library of Faculty of Technical Sciences, Trg Dositeja Obradovića 6, 21000
Note, **N**:	

UNIVERSITY OF NOVI SAD • **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Abstract, **AB**:		This thesis presents research in the field of secure software engineering. Two methods are developed that, when combined, facilitate the integration of software security design analysis into the agile development workflow. The first method is a training framework for creating workshops aimed at teaching software engineers on how to perform security design analysis. The second method is a process that expands on the security design analysis method to facilitate better integration with the needs of the organization. The first method is evaluated through a controlled experiment, while the second method is evaluated through comparative analysis and case study analysis, where the process is tailored and implemented for two different software vendors.	
Accepted by the Scientific Board on, **ASB**:		11.07.2019.	
Defended on, **DE**:			
Defended Board, **DB**:	President:	Branko Milosavljević, PhD, Full Professor, FTN, Novi Sad	
	Member:	Silvia Gilezan, PhD, Full Professor, FTN, Novi Sad	
	Member:	Gordana Milosavljević, PhD, Associate Professor, FTN, Novi Sad	Menthor's signature
	Member:	Žarko Stanisavljević, PhD, Assistant Professor, ETF, Belgrade	
	Member, Mentor:	Goran Sladić, PhD, Associate Professor, FTN, Novi Sad	

Acknowledgements

First of all, I would like to express my sincere gratitude to my mentors Professor Jovanka Pantović and Professor Hugo Torres Vieira, for their unselfish help and experienced guidance. Before starting my PhD studies I had no idea what doing research means. They introduced me to the wonderful world of formal methods and shown me how far one needs to go in order to extract the juice out of his work. Above all, I thank them for always encouraging me to "keep up the good work", and for demonstrating what it takes to be a true scientist and a good person.

To my elementary and high school teachers, and my professors at the university, thank you for making me prepared for doing the Ph.D.

I would like to thank my colleagues at the Chair of Mathematics at Faculty of Technical Sciences for letting me be part of the family and for sharing all the good and the bad that our job carries.

To all of my friends, for making my life happier.

To my family, I owe the most. I thank my "in-law" family Radovan, Jasna, and Maja, for their great support. To my sister Draginja, her husband Bojan and my nieces Jelisaveta and Jovana, to my older brother Mladen and his wife Nina, and to my younger brother Aleksandar, thank you for being a perfect family. To my parents Jovanka and Siniša, for their endless love. To my wife Sanja, and our daughters Tamara and Lenka, for their patience, support, and unquestionable love, thank you for being my world.

Abstract

Distributed software systems have changed the way people communicate, learn and run businesses: almost all aspects of human life have become connected to the internet. The system of interconnected computing devices has numerous positive impacts on everyday life, however, it also raises some concerns, among which are security, accessibility and availability issues.

This thesis investigates problems of formal, mathematically based, representation and analysis of controlled usage and sharing of resources in distributed software systems. The thesis is organized into four chapters. The first chapter provides motivation for our work, and the last concludes the thesis. The second and the third chapters are the core of the thesis, the former addresses controlling information passing and the latter addresses controlling usages of resources.

The second chapter presents a model for confidential name passing, called Confidential π-calculus, abbreviated C_π. This model is a simple fragment of the π-calculus that disables information forwarding directly at the syntax level. We provide an initial investigation of the model by presenting some of its properties, such as the non-forwarding property and the creation of closed domains for channels. We also present examples showing that C_π can be used to model restricted information passing, authentication, closed and open-ended groups. We present an encoding of the (sum-free) π-calculus in C_π and we prove the correctness of the encoding via an operational correspondence result.

The third chapter presents a model of floating authorizations. Our process model introduces floating authorizations as first-class entities, encompassing dimensions of accounting, domain, and delegation. We exploit the language of an already existing process algebra for authorizations, and we adopt a different semantic interpretation so as to capture accounting. We define the semantics of our model in two equivalent ways, using a labeled transition system and a reduction relation. We define error processes as undesired configurations that cannot evolve due to lacking authorizations. The thesis also provides a preliminary investigation of the behavioral semantics of our authorization model, showing some fundamental properties and also informing on the specific nature of floating authorizations.

In the third chapter, we also present a typing discipline that allows to statically single out processes that are not errors and that never evolve into errors, addressing configurations where authorization assignment is not statically prescribed in the system specification. We also develop a refinement of our typing discipline to pave the way for a more efficient type-checking procedure. We show an extended example of a scenario that involves the notion of Bring Your Own License, and we exploit this example to provide insight on a possible application of our model in programming language design.

Key words: distributed systms, cloud computing, micro clouds, edge computing

Rezime

Rasprostranjenost distribuiranih softverskih sistema različitih namena promenila je način na koji ljudi komuniciraju, stiču znanja i vode biznise: skoro svi aspekti ljudskog života postali su povezani sa internetom. Ovaj sistem međusobno povezanih računarskih instanci napravio je veliki pozitivan uticaj na svakodnevni život, od brze i jednostavne komunikacije putem društvenih mreža i računarskih platformi dostupnih putem interneta, do distribuiranih sistema za plaćanja i kriptovaluta zasnovanih na blockchain tehnologijama. Međutim, deljenje informacija, prava pristupa bazama podataka i prava pristupa računarskim platformama, kao i deljenje drugih resursa otvara i nove probleme, među kojima su pitanja bezbednosti, pristupačnosti i dostupnosti. Postoji veliki broj primera u kojima su napadači (krakeri) uspeli da zloupotrebe previde programera koji su razvijali sisteme. Jedan takav skoriji primer je i greška koja je omogućila nepravilno generisanje tokena za pristup ličnim profilima na Facebook-u. Tu grešku je za sada nepoznati napadač uspeo da iskoristi da bi došao do ličnih podataka sa skoro 50 miliona naloga [**?**]. Takvi primeri jasno ukazuju na probleme kontrole deljenja i korišćenja resursa u distribuiranim softverskim sistemima, problemima kojima se ova teza bavi korišćenjem formalnih metoda.

Pouzdanost distribuiranih softverskih sistema može zavisiti od velikog broja faktora i često nije lako čak ni definisati šta se pod pouzdanošću određenog sistema podrazumeva. Jedan od mogućih pristupa kod dizajna i verifikacije takvih sistema je korišćenje formalnih, matematički zasnovanih metoda. Formalne metode predstavljaju tehnike i alate za specifikaciju i verifikaciju kompleksnih (softverskih i hardverskih) sistema zasnovane na matematičkim i logičkim principima. Formalni dizajn obuhvata dve faze: formalnu specifikaciju i verifikaciju. U fazi formalne specifikacije (modeliranja) definiše se sistem koristeći jezik modeliranja, najčešće koristeći preciznu matematičku sintaksu i semantiku. Razvijajući formalnu specifikaciju, uglavnom nastaje i skup teorema koje opisuju osobine tog sistema. U fazi verifikacije, ove teoreme se precizno matematički dokazuju. U konkurentnom računarstvu, neki od poznatih formalnih modela koji se koriste za specifikaciju i verifikaciju osobina sistema su Petrijeve mreže [**?**, **?**], komunicirajući automati sa konačnim brojem stanja [**?**] i procesni računi [**?**, **?**, **?**, **?**].

Cilj ove teze je da predstavi dve specifikacije zasnovane na dva procesna računa koje tretiraju neke aspekte bezbednosti i kontrole pristupa u distribuiranim sistemima. Konačni cilj je stvoriti uslove za bolje razumevanje koncepata izučavanih u ovom radu i omogućiti njihovu kasniju ispravnu implementaciju.

Komunikacija putem distribuiranih sistema, ponekad uključujući interakcije sa nepoznatim i nepouzdanim korisnicima, je postala svakodnevna, a u nekim slučajevima čak i nezaobilazna rutina. U mnogim situacijama razmenjena informacija je privatna i zahteva pažljivo rukovanje i korišćenje. Na primer, osetljivi privatni podaci, kao što su broj kreditne kartice ili adresa, moraju biti otkriveni tokom kupovine putem interneta, ali sa druge strane ove informacije ne bi smele biti dalje deljene od strane korisnika ili aplikacije koji prima informaciju. Ovakvi primeri ukazuju na probleme kontrole deljenja informacija u distribuiranim sistemima. Dakle, deljenje informacija sa trećim licima može dovesti do neželjene diseminacije. Čak i u slučajevima kada se

korisnicima generalno može verovati postoji mogućnost previda koji mogu dovesti do zloupotreba.

Problem privatnosti može i mora biti sagledan sa strane tehnologije ali i prava. Jedan od pionira koji su proučavali privatnost iz obe perspektive je pravnik Alan Westin. On je primetio „da će integracija kontrola privatnosti u nove tehnologije zahtevati snažan napor..." [**?**]. Iako nove tehnologije donose nove pretnje za kontrolu privatnosti, one mogu doneti i nove načine za zaštitu privatnosti [**?**]. Solove [**?**] uvodi taksonomiju i navodi četiri vrste narušavanja privatnosti: sakupljanje informacija, invazija, diseminacija i obrađivanje informacija. Nedovoljna kontrola nad deljenjem informacija u distribuiranim sistemima može biti direktno povezana sa diseminacijom.

Solove daje dalju taksonomiju narušavanja privatnosti putem diseminacije, ali sve ove podvrste uglavnom prepoznaju štetu koja može nastati kod otkrivanja i deljenja osetljivih informacija. Komunikacija među učesnicima je centralni aspekt distribuiranih sistema, a kontrola protoka informacija u takvim sistemima često ima svoje poteškoće. Entiteti u takvim sistemima mogu imati različita prava za manipulaciju određenim informacijama. Na primer, korisnik bankovnog računa ima ovlašćenja da koristi broj kartice za plaćanja putem interneta, može povlačiti određena sredstava sa bankovnog računa, itd. Sa druge strane, kod isplate banka može izvršiti uvid u stanje kako bi proverila da li postoji dovoljno sredstava na računu. Ako se za trenutak fokusiramo na kontrolu protoka informacija, možemo uočiti da bi broj kreditne kartice trebalo da može poslati samo korisnik te kartice, ali ne i banka koja prima tu informaciju. To jest, banka ne bi trebalo da ima prava da prosleđuje informaciju u ovom slučaju. Za narušavanje diseminacije informacija, prosleđivanje može biti prepoznato kao jedna od glavnih meta gde kontrola mora biti uspostavljena.

Ako razmatramo prava koja entitet može imati u odnosu na komunikacioni kanal, možemo razlikovati prava na korišćenje kanala za slanje i čitanje, pravo da se kreira novi kanal i da se pošalje jedan njegov kraj drugom korisniku, prava da se prosleđuju primljena imena

kanala, itd. Davanje prava o prosleđivanju imena kanala svim entitetima apriori može kasnije prouzokovati poteškoće oko kontrole diseminacije, jer u tom slučaju kontrola mora da bude sprovedena u celom sistemu.

Posmatrajmo sada jedan jednostavan primer u kom se poverljivo ime kanala *session* šalje od jednog do drugog korisnika, kao što je onaj naveden u odeljku Introduction. U ovom primeru, korisnik *Alice* kreira novi kanal i šalje jedan njegov kraj korisniku *Bob*. Nakon sinhronizazije u kojoj se razmeni ime kanala, ova dva korisnika mogu napraviti privatnu sesiju na kanalu *session*. Međutim, u našem primeru *Bob* odlučuje da prosledi ime kanala *session* nekom trećem korisniku.

U nekim slučajevima može biti čak i poželjno dati prava prosleđivanja imena kanala nekim korisnicima. Na primer, zadaci mogu biti prosleđivani od nadređenog (eng. master) procesa do potčinjenog (eng. slave) procesa, i tada potčinjeni proces može neprimetno da bude uključen u sesiju. U našem primeru, ova situacija može biti posmatrana kao problematična sa tačke gledišta korisnika *Alice*, jer ona i dalje veruje da drugi kraj kanala, koji ona smatra poverljivim, drži *Bob*. Ako je *session* kanal koji je *Alice* kreirala, kojim se može pristupiti nekim njenim poverljivim podacima, i koji je poslat isključivo korisniku *Bob*, možemo reći da *Bob* ne bi trebalo da stekne mogućnost da ga dalje prosleđuje samo zato što je u nekom trenutku primio ime kanala *session*. U svakom slučaju, možemo napraviti razliku između ova dva slanja, jer prvo slanje je izveo korisnik koji je kreirao kanal (*Alice*), a u drugom je kanal zapravo prosleđen od strane učesnika koji je primio kanal (*Bob*).

Nekoliko formalnih modela je do sada predloženo u svrhu opisivanja restrikovanog deljenja imena, kako bi se postiglo da ime može biti razmenjeno samo u okviru unapred definisanog dela sistema. Takav je i model koji uvodi pojam grupe za imena [**?**] i model koji uvodi pojam skrivanja imena [**?**]. Međutim, u praksi imamo i slučajeve u kojima ne postoji unapred definisan deo sistema u kom poverljiva informacija može biti razmenjena. Na primer, u prethodnom primeru možemo reći

da *Alice* može u nekom trenutku sama da odluči da pošalje ime kanala *session* drugim učenicima. Generalno, privatne informacije nekada moraju biti deljene i u otvorenim sistemima.

Drugi domen koji ova teza obrađuje je izučavanje kontrole prava pristupa u distribuiranim softverskim sistemima. Za početak, možemo primetiti da kontrola prava pristupa računarskim resursima postaje sve važnija, uprkos sve većoj raspoloživosti takvih resursa. Potreba za kontrolom pristupa može biti motivisana mnogim faktorima, kao što su privatnost, bezbednost i ipak postojanje nekog ograničenja kapaciteta. Primeri ograničenog kapaciteta mogu biti direktno povezani sa fizičkim uređajima, kao što su štampači, mobilni telefoni i procesori, jer svi imaju fizički ograničene mogućnosti. Iako neki virtualni uređaji, kao što su deljena memorijska ćelija i web servis, imaju neograničen potencijal, njihova dostupnost je često ograničena.

Privatnost i bezbednost su neki od centralnih problema koji se pojavljuju kod razvoja distribuiranih sistema. Jedan od razloga je taj što distribuirani sistemi postaju sve više heterogeni i kompleksni, a kontrola prava pristupa u takvim sistemima može biti veoma teška. Opravdanje za ovakve tvrdnje možemo naći skoro svakodnevno, već pomenuti primer greške na Facebook-u je samo jedan u nizu. Takvi primeri su prouzrokovali milionske gubitke kompanija, ali još važnije, sigurnost i privatnost korisnika je u takvim situacijama bila izložena opasnosti. Formalni modeli i verifikacije mogu biti korak bliže ka pouzdanijim distribuiranim softverskim sistemima [**?**].

Različite metode za kontrolu prava pristupa u distribuiranim softverskim sistemima razvijane su tokom godina. Njihov razvoj pratio je stalne promene u strukturi i veličini sistema. Za male sisteme, i za sisteme sa unapred definisanim brojem učesnika, kontrola prava pristupa resursima obično se postiže korišćenjem lista za kontrolu pristupa (eng. access control lists - ACL). ACL metoda koristi liste sa pravima koje su dodeljene resursima. Pravo pristupa resursu može biti odobreno samo korisniku koji je naveden kao subjekat sa odgovarajućim pravom pristupa na listi datog resursa.

Iako ACL metod daje prirodan način za kontrolu prava pristupa, u velikim sistemima koji su dinamični po pitanju broja i sastava učesnika ovaj metod postaje težak za implementaciju. Razlog za to je što u ACL metodi svaka lista čuva podatke o svakom korisniku individualno, a to može predstavljati veliki trošak pri održavanju sistema. Na primer, posmatrajmo aplikaciju kao što je Facebook, koju koristi preko milijardu korisnika. Imati liste korisnika koji mogu da pristupe resursima, kao što su fotografije ili postovi svakog korisnika, može postati nepraktično.

Upravljanje pristupom na osnovu uloga (eng. role-based access control method - RBAC) [**?**] je uvedeno kao alternativa ACL metodi. RBAC metoda definiše skup uloga i svakom korisniku dodeljuje se jedna ili više uloga. Na primer, da bi sistem korisniku dozvolio ili odbio pristup fotografiji drugog korisnika na Facebook-u, ne mora se oslanjati na njegov identitet direktno. Praktičnije rešenje je proveriti da li korisnik koji pokušava da pristupi fotografiji ima ulogu „prijatelja" sa vlasnikom fotografije. Pored svih prednosti (i mana) koje RBAC metoda ima u poređenju sa ACL metodom, ona i dalje ima nedostatak da mora postojati centralni mehanizam za izdavanje i proveravanje uloga korisnika.

Upravljanje pristupom na osnovu ključa (eng. capability-based method for access control) [**?**] je metoda koja je više prilagođena decentralizovanim sistemima. U ovoj metodi, reference koje se ne mogu kopirati kreira i izdaje centralni mehanizam. Jednom izdata referenca ostaje kod korisnika i proverava se samo kada korisnik želi da pristupi resursu. Dakle, u ovoj metodi centralni mehanizam ne mora da drži informacije o kontroli pristupa za svakog korisnika pojedinačno, dovoljno je da proverava validnost referenci (ključeva) samo kada je to potrebno. Takođe, ove reference mogu biti delegirane između dva učesnika, bez potrebe da se o tome obavesti centralni mehanizam za kontolu pristupa.

Još jedan domen koji obuhvata slične principe kao i poslednja navedena metoda za upravljanje pristupom je domen licenci: korisnik može

upotrebiti određenu aplikaciju samo pod uslovom da poseduje odgovarajuću licencu. U ovom domenu takođe možemo naći pojam eksplicitne delegacije. Na primer, korisnik koji želi da uposli aplikaciju na računarskoj platformi dostupnoj putem interneta može delegirati licencu za tu aplikaciju koju već poseduje. Taj pojam poznat je pod nazivom Bring Your Own License [**?**] (BYOL). Posebna vrsta licenci kao što su licence za konkurentnu upotrebu (eng. concurrent use licenses) nudi dodatnu fleksibilnost kod korišćenja [**?**]. Kao primer, posmatrajmo jednu kompaniju koja koristi aplikaciju i koja poseduje određeni broj licenci potrebnih za korišćenje te aplikacije. U slučaju licenci za konkurentnu upotrebu, licence mogu biti dostupne svim korisnicima u okviru domena date kompanije, ali broj licenci određuje gornju granicu za broj korisnika koji mogu koristiti aplikaciju u bilo kom trenutku [**?**].

U ovoj tezi istražujemo probleme formalnog, matematički zasnovanog, modeliranja i analize kontrolisanog korišćenja i deljenja resursa u distribuiranim softverskim sistemima. Teza je organizovana u četiri poglavlja.

Prvo poglavlje daje motivaciju za razvoj modela uvedenih u drugom i trećem poglavlju teze.

Drugo poglavlje ove teze daje jedan novi pristup za proučavanje prvog problema koji smo do sada naveli: ograničene diseminacije poverljivih informacija. U ovom poglavlju uvodimo formalni model koji ograničava komunikacije koje se mogu okarakterisati kao prosleđivanje. U tu svrhu predstavljen je račun nazvan *Confidential π-calculus*, ili skraćeno C_π. Ovaj račun predstavlja jedan fragment čuvenog Milnerovog π-računa [**?**], koji direktno u sintaksi onemogućava prosleđivanje primljenih imena. Jedini resursi u našem modelu su imena kanala, i mi tretiramo imena kanala kao poverljive informacije. Glavna razlika u poređenju sa originalnim π-računom je ta što u C_π-računu jednom primljena imena kanala kasnije nije moguće poslati. Ovo poglavlje teze se oslanja na publikovani rad

1. I. Prokić. The Cpi-calculus: a model for confidential name passing. In M. Bartoletti, L. Henrio, A. Mavridou, and A. Scalas, editors, *Proceedings 12th Interaction and Concurrency Experience, ICE 2019, Copenhagen, Denmark, 20-21 June 2019*, volume 304 of *Electronic Proceedings in Theoretical Computer Science*, pages 115–136. Open Publishing Association, 2019.

ali ga dopunjava i proširuje. Takođe, ovde uvodimo novo pojednostavljeno kodiranje iz π-računa u C_π-račun i predstavljamo kompletan dokaz operacione korespondencije za ovde uvedeno kodiranje. Doprinosi ovog poglavlja u tezi su sledeći:

- Uvođenje novog, jednostavnog fragmenta π-računa koji nam omogućava da predstavimo komuniciranje poverljivih imena ograničavanjem mogućnosti prosleđivanja imena. Činjenica da je uvedeni model fragment uveliko izučavanog π-računa, daje nam mogućnost da iskoristimo već razvijene teorijske rezultate koji postoje za π-račun.

- Uvođenje definicije osobine neprosleđivanja i, kao provera dobre zasnovanosti, pokazivanje da svi procesi iz našeg C_π-računa zadovoljavaju ovu osobinu.

- Koristeći jaku bisimulaciju, bihevioralnu ekvivalenciju iz π-računa, pokazan je jedan bihevioralni identitet koji potvrđuje da u našem računu možemo direktno predstaviti kreiranje zatvorenih domena za kanale.

- Data je detaljna diskusija o ekspresivnosti C_π-računa na nekoliko proširenih primera, koji uključuju reprezentaciju kreiranja zatvorenih domena za kanale, autentikacije, zatvorenih i otvorenih grupa, od kojih svi mogu biti direktno predstavljeni u našem modelu.

- Uvedeno je novo kodiranje π-računa u C_π-račun, čime je pokazano da je naš račun, iako predstavlja tek fragment π-računa koji razmatra samo deo njegove sintakse, podjednako ekspresivan kao i

π-račun. Takođe, u ovom poglavlju dat je detaljan dokaz operacione korespondencije za uvedeno kodiranje.

Centralni pojam svih formalnih modela za konkurentne i distribuirane sisteme je proces. Proces označava entitet koji može da komunicira sa drugim takvim entitetima koristeći zajedničke komunikacione kanale. Neke od prvih i najviše izučavanih procesnih algebri su Milnerov račun komunikacionih sistema (eng. Calculus of Communicating Systems - CCS) [**?**] i Hoareov račun komunikacionih sekvencijalnih procesa (eng. Communicating Sequential Processes - CSP) [**?**]. Napomenimo da je CSP poslužio kao osnovni model za programski jezik *Go* koji je razvio Google. Za sveobuhvatniji pregled istorije razvoja procesnih algebri pogledati [**?**].

Milnerov CCS-račun je jedan od prvih koji je formalno izučavao konkurentne sisteme. Ovaj račun uvodi pojmove paralelne kompozicije, sinhronizacije slanja i primanja na istom imenu, kreiranja privatnih imena i sumacije (izbora). U ovoj tezi operator sumacije nije razmatran, ali verujemo da bi dodavanje ovog operatora moglo da se uradi na uobičajen način. U CCS-računu možemo definisati proces $Alice \mid Bob$ koji označava dva konkurentna potprocesa $Alice$ i Bob, spojena operatorom paralelne kompozicije. Dva konkurentna procesa mogu da se sinhronizuju putem zajedničkog kanala. Recimo, u procesu

$$\overline{chn}.Alice \mid chn.Bob$$

proces na levoj strani $\overline{chn}.Alice$ može da izvede akciju slanja na kanalu chn, dok proces na desnoj strani može da izvede (dualnu) akciju primanja na istom kanalu. Nakon sinhronizacije početni proces se svodi na $Alice \mid Bob$. U CCS-računu procesi mogu i da kreiraju nova imena kanala, čime se modeluje stvaranje privatnih kanala koji nisu dostupni drugim procesima. U procesu

$$((\nu session)Alice) \mid Bob$$

ime kanala $session$ je poznato samo procesu $Alice$ i može biti korišćeno samo za sinhronizacije unutar tog procesa, dok proces Bob

nema nikakvu informaciju o postojanju tog kanala. Ono što CCS-račun ne može da predstavi direktno jeste mobilnost kanala.

Tamo gde je Milner stao sa CCS-računom, nastavio je sa π-računom, koji proširuje CCS da bi dozvolio mobilnost komunikacionih kanala. U π-računu procesi u toku sinhronizacije na kanalu mogu da razmenjuju imena kanala, time stvarajući nove konekcije među sobom. Nekoliko programskih jezika inspirisano je ovim modelom [**?**, **?**, **?**, **?**, **?**, **?**].

U π-računu, možemo definisati proces

$$chn!session.Alice \mid chn?x.Bob$$

gde na levoj strani paralelne kompozicije imamo proces koji je spreman da šalje ime kanala *session* putem kanala *chn*, dok na desnoj strani imamo proces koji je spreman da primi bilo koje ime kanala na kanalu *chn*, a zatim da ime x (koje se još zove i „placeholder") unutar procesa *Bob* bude zamenjeno primljenim. Ovaj mehanizam daje novu dimenziju kada se kombinuje sa kreiranjem novih kanala, jer sada kreirani kanali mogu biti razmenjeni među procesima, čime se mogu stvarati privatne konekcije. Ovo je ujedno i poslednji sastojak koji nam je trebao da bismo u π-računu modelovali primer sa prosleđivanjem imena kanala koji smo ranije spominjali:

$$((\nu session)chn!session.Alice) \mid chn?x.forward!x.Bob'$$

U ovom procesu kanal *session* je poznat samo potprocesu sa leve strane paralelne kompozicije. Međutim, nakon sinhronizacije sa potprocesom sa desne strane, početna konfiguracija evoluira u

$$(\nu session)(Alice \mid forward!session.Bob'')$$

gde je ime privatnog kanala *session* sada poznato i desnom potprocesu (to jest, Bob'' predstavlja proces koji se dobije od procesa Bob' kada sva pojavljivanja imena x zamenimo imenom *session*). Dakle, u π-računu domen privatnog imena (što predstavlja deo sistema gde je ime

poznato) se može uvećati nakon sinhronizacije. Ako pretpostavimo da paralelno postoji i treći aktivni process

$$(\nu session)(Alice \mid forward!session.Bob'') \mid forward?y.Carol$$

onda proces koji obuhvata Bob'' može sada proslediti ime kanala *session* tom trećem procesu putem kanala *forward*, a da o tome prethodno nije obavestio *Alice*. Ova diseminacija imena može dovesti do situacije u kojoj je privatnost *Alice* kompromitovana.

C_π-račun diskvalifikuje osobinu prosleđivanja, te stoga $chn?x.forward!x.Bob'$ nije C_π proces. Formalno, naš račun razlikuje imena kanala i imena promenljivih koje se pojavljuju u prefiksu primanja (eng. placeholder). Mi uvodimo dva disjunktna skupa imena, jedan označen sa $\mathcal{C}$ koji čine imena kanala, i drugi označen sa $\mathcal{V}$ koji čine imena promenljivih. Ova distinkcija je iskorišćena kod definisanja jezika našeg modela, jedino imena iz skupa $\mathcal{C}$ mogu biti navedena kao imena za slanje u prefiksu koji definiše ovu akciju. Ovakvo sintaksno ograničenje samo po sebi ne daje uopšteno ograničenje da se imena kanala, koja su posmatrana kao poverljiva informacija, ne mogu razmenjivati, niti ograničava deo sistema u kom ime može biti primljeno. Ono što C_π postiže zapravo je lokalizacija dela sistema koji može poslati ime kanala, a to je onaj deo gde je kanal prvobitno kreiran. Ukoliko je neophodno uspostaviti kontrolu nad slanjem imena nekog kanala, u C_π-računu je dovoljno skoncentrisati se na deo sistema gde je kanal kreiran, dok bi, recimo, u π-računu bilo neophodno kontrolu uspostaviti nad čitavim delom sistema koji zna za dato ime.

Ono što je posledica specifičnosti C_π-računa je to da možemo razlikovati dva nivoa ovlašćenja koja proces može imati u odnosu na neki kanal. Proces koji kreira kanal ima ovlašćenja da komunicira putem kanala, ali takođe može i da pošalje ime kanala drugim procesima. Proces koji u nekom trenutku primi ime kanala stiče pravo da komunicira putem tog kanala, ali ne i da dalje prosleđuje ime tog kanala. Prvi tip procesa u ovoj tezi je nazvan administrator kanala, a drugi korisnik kanala. Svaki administrator je istovremeno i korisnik, ali korisnik ne mora biti i administrator. Još jedna posledica lokalizacije

dela sistema u kom se ime datog kanala može poslati u tezi je iskorišćena i da pokaže kako C_π-račun može biti iskorišćen za modelovanje autentikacije. Naime, sama mogućnost slanja imena nekog kanala zapravo pripada samo administratoru kanala, a onom procesu koji prima to ime zapravo govori sa kojim procesom u tom trenutku komunicira (sa administratorom tog kanala).

Restrikcija koju C_π-račun pravi u odnosu na π-račun zapravo suštinski ne utiče na ekspresivnu moć, a to je i dokazano u samoj tezi. Sama ideja reprezentacije prosleđivanja u C_π-računu je izdvajanje procesa koji bi bili zaduženi isključivo za slanje određenog imena kanala. Drugi procesi bi, ukoliko žele da pošalju neko ime kanala, zapravo umesto slanja samog imena prvo kontaktirali odgovarajući izdvojeni proces koji bi izvršio slanje umesto njih. Ova ideja je u tezi formalizovana u kodiranju π-računa u C_π-račun.

Veliki broj teorijskih istraživanja konkurentnih i distribuiranih sistema direktno je povezan sa π-računom. Mnogi radovi koriste π-račun kao osnovni i proširuju njegovu sintaksu kako bi stekli odgovarajući nivo apstrakcije da modeluju poliadične komunikacije [**?**, **?**], komunikacije višeg reda [**?**], distribuirane sisteme [**?**], sigurnost i privatnost [**?**, **?**, **?**, **?**, **?**, **?**], i mnoge druge aspekte, uključujući i kontrolu korišćenja resursa koju razmatramo u trećem poglavlju ove teze. Sa druge strane, deo istraživača je koristio sužavanje sintakse π-računa kako bi modelovali asinhrone komunikacije [**?**, **?**], unutrašnju mobilnost [**?**], i lokalizaciju [**?**], a naš C_π-račun svakako spada u ovu kategoriju.

Treće poglavlje predstavlja formalni model za izučavanje kontrole pristupa resursima u distribuiranim softverskim sistemima. Ovaj račun u apstraktnom smislu modeluje „capabilities" metodu za kontrolu pristupa, ali takođe i licence za konkurentnu upotrebu, uvodeći pojam deljene autorizacije. Autorizacije se mogu definisati kao funkcije koje određuju prava i privilegije u odnosu na neki resurs. Kao i u

drugom poglavlju, i ovde nam je fokus na sistemima kod kojih je komunikacija centralni pojam, tako da su jedini resursi koje ovde razmatramo zapravo imena komunikacionih kanala. Dakle, autorizacija definiše pravo da se koristi određeni komunikacioni kanal. Deljene autorizacije, koje posmatramo u ovom modelu, definišu prava da se kanal koristi konkurentno. Ovo zapravo znači da jedna autorizacija može biti dostupna većem broju korisnika, ali da je u svakom trenutku može koristiti najviše jedan korisnik. U ovom modelu koristimo destilovane osobine deljenih autorizacija: domen, koji definiše deo sistema u kome je autorizacija implicitno dostupna; brojanje, koje definiše kapacitet; delegacija, koja definiše slanje i primanje samih autorizacija.

Model predstavljen u trećem poglavlju je zapravo ekstenzija π-računa [**?**], koji se direktno oslanja na pretodno razvijeni račun sa autorizacijama [**?**, **?**]. Iz računa sa autorizacijama [**?**] preuzeti su sintaksni konstrukti za domen i delegaciju autorizacija. U semantičkom smislu, naš model modifikuje samo značenje domena autorizacije, kako bi dobili mogućnost da obuhvatimo princip o brojanju autorizacija koji proističe iz prirode deljenih autorizacija izučavanih ovde. Treće poglavlje sistematično iznosi rezultate koji su prethodno predstavljeni u publikovanim radovima:

1. J. Pantović, I. Prokić, and H. T. Vieira. A calculus for modeling floating authorizations. In C. Baier and L. Caires, editors, *Formal Techniques for Distributed Objects, Components, and Systems - 38th IFIP WG 6.1 International Conference, FORTE 2018, Held as Part of the 13th International Federated Conference on Distributed Computing Techniques, DisCoTec 2018, Madrid, Spain, June 18-21, 2018, Proceedings*, volume 10854 of *Lecture Notes in Computer Science*, pages 101–120. Springer, 2018.

2. I. Prokić, J. Pantović, and H. T. Vieira. A calculus for modeling floating authorizations. *Journal of Logical and Algebraic Methods in Programming*, 107:136 – 174, 2019.

Glavni doprinosi ukupnog rada na modelu koji uvodi deljene autorizacije u π-račun u trećem poglavlju ove teze su sledeći:

- Definisanje novog formalnog računa koji modeluje već spomenute pojmove domena, deljenih resursa, brojanja i delegacije, uvođenjem pojma deljene autorizacije.

- Izučavanje bihevioralne semantike ovog modela. Izvedena bihevioralna karakterizacija pokazuje specifičnu prirodu deljenih autorizacija, naročito odnos između konstrukta za domen autorizacije i konstrukta za paralelnu kompoziciju, koja reflektuje gore spomenuti princip brojanja.

- Uvođenje tipskog sistema koji omogućava izdvajanje procesa koji autorizovano koriste svoje kanale, čak i u prisustvu autorizacija koje su obezbeđene od strane konteksta. Dokazivanje rezultata koji pokazuju da dobro tipiziran process ne samo da uvek koristi svoje kanale autorizovano, već to takođe važi i za sve njegove moguće evolucije.

- Poboljšanje efikasnosti algoritma za proveru tipa uvođenjem drugog tipskog sistema, za koji je pokazano rezultatom tipske korespondencije da je ekvivalentan sa prvim tipskim sistemom.

- Prikazivanje proširenog primera inspirisanog pojmom *Bring Your Own License* iz domena licenci, koji detaljnije opisuje uvedeni model i koji povezuje model sa njegovim mogućim aplikacijama.

- Na osnovu pomenutog proširenog primera pokazan je jedan konkretan pravac za primenu definisanog modela u programskim jezicima. Dat je primer koji posmatra jednu moguću ekstenziju programskog jezika Go[1].

[1] https://golang.org

Model sa deljenim autorizacijama uspostavlja dodatni nivo kontrole korišćenja kanala u odnosu na π-račun. U našem modelu, nije dovoljno da proces ima pristup kanalu, već dodatno mora imati i autorizaciju za korišćenje tog kanala. Sama sintaksa π-računa proširena je sa konstruktima za autorizacije i njihovo delegiranje među procesima. Na primer, proces

$$(license)(Alice \mid Bob)$$

definiše da je jedna autorizacija za korišćenje kanala *license* dostupna procesima *Alice* i *Bob*. Ako, recimo, proces *Bob* prvi započne komunikaciju na kanalu *license* onda konfiguracija data gore postaje

$$Alice \mid (license)LicensedBob$$

gde autorizacija *license* više nije dostupna za *Alice*. Autorizacije mogu biti razmenjene u komunikaciji. Na primer, u

$$(license)(auth)auth\langle license\rangle.UnlicensedBob \mid (auth)auth(license).LicensedCarol$$

proces na levoj strani ima autorizaciju da koristi kanale *auth* i *license*, a prefiks definiše akciju slanja autorizacije za *license* putem kanala *auth*. Sa desne strane, proces može da primi autorizaciju za *license* na kanalu *auth*, i za tu akciju ima odgovarajuću autorizaciju. Nakon sinhronizacije dva procesa, dobijemo

$$(auth)\,UnlicensedBob \mid (auth)(license)LicensedCarol$$

gde autorizacija za *license* prelazi sa leve na desnu stranu.

Kao što smo videli u prethodnom primeru, autorizacije zapravo omogućavaju (ili u nedostatku istih, onemogućavaju) komunikacije na kanalima. Ovo važi ne samo za komunikacije u kojima se razmenjuju autorizacije, već i za komunikacije u kojima se razmenjuju imena kanala. Na primer,

$$(comm)comm!license.Alice \mid (comm)comm?x.Dylan$$

predstavlja proces u kome ime kanala *license* poslato na kanalu *comm* od potprocesa sa leve strane, može biti primljeno u potprocesu sa desne strane, jer za obe akcije postoje odgovarajuće autorizacije. Sa druge strane, sinhronizacija u procesu

$$(comm)(comm!license.Alice \mid comm?x.Dylan)$$

nije moguća jer za akcije slanja i primanja postoji samo jedna autorizacija, dok su potrebne dve. U ovoj tezi, ovakvi procesi, koji ne mogu da sinhronizuju svoje dualne akcije zbog nedostatka odgovorajućih autorizacija nazivaju se greškama.

Da bi izdvojili procese koji nisu greške i koji ni u jednoj od mogućih evolucija ne postaju greške, u tezi je predstavljen tipski sistem. Tipski sistem se sastoji od dodele tipova imenima i tipskih pravila, koja definišu uslove koje proces koji se proverava mora zadovoljiti. Ukoliko proces može da prođe definisanu tipsku proveru onda je on „bezbedan", to jest, nije greška i ne svodi se na grešku. Tipovi koje mi ovde dodeljujemo imenima zapravo govore o imenima koja mogu biti bezbedno komunicirana na kanalima. Na primer, posmatrajmo process

$$(exam)(minitest)(alice)alice?x.x!value.0$$

koji može da primi ime kanala i da zatim pošalje *value* na primljenom kanalu. Primanje na *alice* je autorizovano direktno jer je odgovarajuća autorizacija prisutna. Sa druge strane, kasnije slanje je autorizovano samo za imena *exam* i *minitest*. Ako možemo da osiguramo da na kanalu *alice* samo imena *exam* i *minitest* mogu biti komunicirana, tada je ovaj process bezbedan. Stoga, imenu *alice* dodeljujemo tip $\{alice\}(\{exam, minitest\}(\emptyset))$, i to obeležavamo sa

$$alice : \{alice\}(\{exam, minitest\}(\emptyset))$$

kako bismo označili da je *alice* finalno ime (uporediti sa tipom od x datim dole), i da kanal može biti korišćen isključivo za komuniciranje imena *exam* i *minitest*. Poslednja informacija u tipu govori da *exam*

i *minitest* ne mogu biti korišćeni za komunikacije (označeno sa $\emptyset$). Sa druge strane, tip koji bismo dodelili promenljivoj u ovom procesu je $x : \{exam, minitest\}(\emptyset)$, jer x može biti zamenjeno imenima *exam* ili *minitest*, za koje onda treba obezbediti autorizacije. Dakle, za samo ime x autorizacija nije prisutna u procesu u kom se nalazi prefiks, ali autorizacije za dve moguće zamene imena jesu, zbog čega ovakve indirektne autorizacije zovemo kontekstualne autorizacije. Tipski sistem predstavljen u ovoj tezi tretira i direktne i kontekstualne autorizacije. Tipske pretpostavke skupljaju se u tipsko okruženje, obično obeleženo sa Δ, i u odnosu na takvo okuženje vrši se provera procesa pomoću pravila koja se definišu za svaki sintatički konstrukt pojedinačno. Tipsko tvrđenje, koje je oblika $\Delta \vdash_\rho P$, govori da proces P koristi svoje kanale kako je to propisano u tipskom okruženju Δ i da proces poseduje dovoljno autorizacija ako bi bio smešten u kontekst koji bi mu obezbedio dodatne autorizacije navedene u multiskupu ρ. Ove ideje su takođe formalizovane u trećem poglavlju teze.

Četvrto poglavlje sadrži sažetak postignutih rezultata kandidata, pregled literature i razmatra pravce daljih istraživanja.

Ključ reči: distributed systms, cloud computing, micro clouds, edge computing

Table of Contents

List of Figures

List of Tables

List of Abbreviations

EPEI	Every Part Every Interval
TPS	Toyota Production System
TSP	Transport Service Provider

Chapter 1

Introduction

1.1 Distributed systems

1.1.1 Cloud computing

1.1.2 Peer-to-peer networks

1.1.3 Mobile computing

1.2 Distributed computing

1.2.1 Big Data

1.2.2 Microservices

1.3 Problem Statement

1.4 Research Hypotheses, and Goalst

1.5 Structure of the thesis

Chapter 2

Research review

2.1 Platform models

2.2 Nodes organization

2.3 Task offloading

Chapter 3

Micro clouds and edge computing as a service

3.1 Formal model

3.2 Configurable Model Structure

3.3 Applications Model

3.3.1 Packaging

3.3.2 Execution

3.4 Results

Chapter 4

Conclusion

4.1 Future work

Bibliography