UNIVERZITET U BEOGRADU

ELEKTROTEHNIČKI FAKULTET

Nikola S. Bežanić

MODEL IMPLEMENTACIJE SERVISNO ORIJENTISANIH MREŽA PAMETNIH PRETVARAČA

Doktorska disertacija

Beograd, 2015.

UNIVERSITY OF BELGRADE

SCHOOL OF ELECTRICAL ENGINEERING

Nikola S. Bežanić

IMPLEMENTATION MODEL FOR SERVICE ORIENTED SMART TRANSDUCERS NETWORKS

Doctoral Dissertation

Belgrade, 2015.

PODACI O MENTORU I ČLANOVIMA KOMISIJE

Mentor:

Doc. dr Ivan Popović,
Univerzitet u Beogradu - Elektrotehnički fakultet

Članovi komisije:

Prof. dr Vujo Drndarević,
Univerzitet u Beogradu - Elektrotehnički fakultet

Prof. dr Lazar Saranovac,
Univerzitet u Beogradu - Elektrotehnički fakultet

Prof. dr Miroslav Lutovac,
Univerzitet Singidunum - Fakultet za informatiku i računarstvo

Doc. dr Aleksandar Rakić,
Univerzitet u Beogradu - Elektrotehnički fakultet

Datum odbrane: _____

MODEL IMPLEMENTACIJE SERVISNO ORIJENTISANIH MREŽA PAMETNIH PRETVARAČA

Rezime — Pametni pretvarač je uređaj koji pored pretvaračkih elemenata i pratećih kola za obradu signala, poseduje i lokalnu inteligenciju i mogućnost komunikacije. Veliki broj načina na koji se takav uređaj može realizovati, dovodi do nekompatibilnosti i niskog nivoa interoperabilnosti između uređaja različitih proizvođača, zbog čega je uveden standardni model pametnog pretvarača definisan familijom standarda IEEE 1451. Dodatno unapređenje interoperabilnosti se postiže nadogradnjom standarda, kroz uvođenje posebnih Web servisa za komunikaciju sa pametnim pretvaračima.

Analiza postojećih rešenja je pokazala da i pored uvedene standardizacije ne postoji opšti model mrežne konfiguracije i komunikacije. Zbog toga je u ovoj tezi razvijen model implementacije koji omogućava uniformnu konfiguraciju svih mrežnih čvorova od strane centralnog serverskog čvora i koji uvodi dva obrasca komunikacije: (i) komunikaciju posredstvom centralnog servera i (ii) obrazac zasnovan na upotrebi servisnih agenata.

Servisni agenti su uvedeni kao aktivne komponente, zadužene za transfer podataka između pasivnih komponenata sistema kojima se pristupa putem servisnih interfejsa. Servisni agenti se mogu nalaziti na bilo kom mrežnom čvoru, čime se eliminiše centralizovani model komunikacije i omogućava proizvoljna mrežna topologija. Za potrebe integracije entiteta koji nisu konkretni pametni pretvarači, poput algoritama za obradu podataka i ulazno/izlaznih uređaja, predstavljen je koncept virtuelnog pretvaračkog modula. Dati mrežni entiteti u formi virtuelnog pretvaračkog modula se na nivou interfejsa vide isto kao i konkretni pametni pretvarači, što omogućava uniforman pristup od strane centralnog menadžerskog čvora i servisnih agenata.

Verifikacija modela je data putem studija slučaja mreža za osmatranje parametara okoline, procenu uslova za pojavu leda, predikciju signala upotrebom neuralnih mreža i kontrole temperature sušare. Zaključeno je da predloženi model ima praktičnu primenu, pri čemu je pokriven širok spektar mreža pametnih pretvarača, podržava upotrebu i drugih funkcionalnosti koje nisu date IEEE 1451 specifikacijom i različite komunikacione protokole, što ga čini pogodnim za dalji razvoj servisno orijentisanih mreža pametnih pretvarača.

Ključne reči: servisno orijentisana arhitektura, pametni pretvarači, IEEE 1451, distribuirani merni sistemi
Naučna oblast: tehničke nauke, elektrotehnika
Uža naučna oblast: elektronika
UDK broj: 621.3

IMPLEMENTATION MODEL FOR SERVICE ORIENTED SMART TRANSDUCERS NETWORKS

Abstract — Smart transducer is a device possessing the local intelligence and communication capability, apart from the basic transducer elements and accompanying signal processing circuits. A large number of different design possibilities leads to the incompatibility and poor interoperability between devices of different manufacturers, which are solved by the smart transducer model established by the IEEE 1451 family of standards. Additional improvement of interoperability is achieved by an upgrade of the standard, through introduction of the customized Web services for communication with smart transducers.

An analysis of existing solutions showed that the general model of network level configuration and communication, actually, does not exist, although the standardized smart transducer model does. Because of that, this work establishes an implementation model which enables uniform configuration of all network nodes by the central server node, and introduces the following two communication patterns: (i) communication over the central server intermediary and (ii) communication based on the service agents deployment.

Service agents are introduced as the active components that transfer data between the passive system components accessed through service interfaces and can be deployed to an arbitrary network node, which eliminates the centralized communication model and enables arbitrary network topology. In order to integrate the entities which are not physical smart transducers in the network, like processing algorithms and input/output devices, concept of the virtual transducer interface module is introduced. The given network entities in form of the virtual transducer interface module are seen at the interface level the same as the physical smart transducers, and are accessed uniformlly by the central manager node and service agents.

Model verification is given through the case studies of networks for environmental monitoring, ice formation condition estimation, neural network signal sequence prediction, and dryer chamber temperature control. It is concluded that the proposed model is applicable to a wide range of smart transducers networks, it enables functionalities other than IEEE 1451 and different communication protocols, which makes it suitable for further development of the service oriented smart transducers networks.

Keywords: service oriented architecture, smart transducers, IEEE 1451, distributed measurement systems
Scientific area: technical science, electrical engineering
Specific scientific area: electronics
UDK number: 621.3

SADRŽAJ

1. UVOD

Upotreba pretvarača je česta u savremenom okruženju u merno-upravljačkim aplikacijama, medicini, praćenju parametara okoline, avio industriji i drugim oblastima. Pri tome, pretvarač može biti senzor ili aktuator i koristi se za konverziju energije iz jednog domena u drugi, pri čemu senzor konvertuje neku fizičku veličinu (npr. temperaturu, pritisak) u električni signal, dok aktuator pretvara električni signal u neku fizičku akciju. Osim za praćenje fizičkih veličina, postoje i drugi tipovi senzora poput onih za detekciju i određivanje koncentracije hemijskih jedinjenja.

Prilikom merenja neke fizičke veličine, senzor se tipično spreže sa kolima za obradu analognog signala, a zatim se tako dobijeni analogni signal prebacuje u digitalni domen putem analogno-digitalnog konvertora. Novodobijeni digitalni podatak se može obrađivati pomoću procesorske jedinice, čuvati u memoriji računara i učiniti dostupnim drugim uređajima upotrebom digitalnih komunikacionih interfejsa. Uređaj koji integriše pretvarače, kola za obradu i pojačanje signala, analogno-digitalne i digitalno-analogne konvertore, procesorsku jedinicu i komunikacioni interfejs, može se smatrati pametnim pretvaračem [1]. Implementacija pametnih pretvarača na bazi mikrokontrolera omogućava integraciju svih potrebnih komponenti, čime se obezbeđuje i povezivanje na digitalnu komunikacionu mrežu, pri čemu je razmena podataka sa pametnim pretvaračem olakšana uvođenjem standardnih bežičnih interfejsa i žične serijske komunikacije. Ipak, zbog velikog broja načina na koji se može realizovati pametni pretvarač i velikog broja komunikacionih protokola, zajedničkom inicijativom američkog instituta *National Institute of Standards and Technology* (NIST), udruženja inženjera *Institute of Electrical and Electronics Engineers* (IEEE) i industrije, definisana je familija standarda IEEE 1451. Standard omogućava proizvodnju pametnih pretvarača na jedinstven način, pri čemu se ne moraju izrađivati posebni uređaji za svaku od mreža dostupnih na tržištu.

Nasuprot standardizaciji pametnih pretvarača, u cilju unapređena poslovanja preduzeća primenom IT tehnologija, uvedena je arhitektura sistema koja se zasniva na servisno orijentisanom dizajnu. Ovakva arhitektura je u stranoj literaturi poznata pod nazivom *Service Oriented Architecture* (SOA). Jedan od načina za realizaciju SOA sistema jeste korišćenje tehnologije Web servisa, koji se praktično mogu implementirati na proizvoljnim platformama. Realizacija poslovne logike u vidu servisa kojima se pristupa unutar organizacije ili putem spoljne mreže daje mogućnost projektovanja distribuiranih sistema sa jasno definisanim interfejsima. Interfejsi se mogu definisati tako da budu apstraktni i nezavisni od načina implementacije, što omogućava platformski nezavisnu mrežnu integraciju. Dizajn sistema prema principima SOA predstavlja oblast sa širokim spektrom mogućnosti, pri čemu ne postoji standardna definicija optimalne arhitekture. Za svaki pojedinačan slučaj je potrebno razmotriti potrebe sistema i shodno tome definisati skup potrebnih servisa i izabrati pogodan način njihove realizacije. Enkapsulacija poslovne logike u servisne komponente sa jasno definisanim interfejsima omogućava efikasnu mrežnu integraciju heterogenih uređaja u okviru distribuiranog sistema.

U ovoj tezi će biti analiziran model implementacije mreža pametnih pretvarača prema SOA konceptu, sa pretpostavkom da "poslovnu logiku" Web servisa čine uobičajene operacije pametnog pretvarača. Takođe, u predloženom servisno orijentisanom okruženju, biće analizirane mogućnosti mrežne integracije entiteta poput algoritama i displeja za prikaz rezultata merenja, kao i mogućnosti efikasne konfiguracije mreže. Predloženi model omogućava dislokaciju kompleksnih funkcionalnosti i servisa između mrežnih čvorova, što vodi ka smanjenim softverskim/hardverskim zahtevima prilikom realizacije mrežnih čvorova na bazi namenskih računarskih sistema. Predložena rešenja će biti data i u formi prototipa, kojima se dokazuje njihova izvodljivost i vide specifičnosti praktične primene.

U okviru teze, prvobitno je dat pregled stanja u oblasti, pri čemu je opisana standardizacija i postojeći modeli pametnih pretvarača, kao i standardizacija aplikacija na bazi tih modela. Zatim je dat opis dva nova pristupa u modeliranju servisno orijentisanih mreža pametnih pretvarača, pri čemu se prvi zasniva na razmeni podataka u zvezdastoj topologiji, dok drugi obezbeđuje proizvoljnu mrežnu topologiju. U okviru datih modela je rešen problem standardnog načina konfiguracije mreže, dislokacije funkcionalnosti i vremenske sinhronizacije poslova različitih mrežnih čvorova, dok je verifikacija modela data putem pojedinačnih studija slučaja sa prikazom rezultata. Na kraju je dat osvrt na ostvarene rezultate u okviru teze i autorova predviđanja daljih tokova istraživanja.

Glavni naučni doprinos ove doktorske teze se sastoji iz sledećih elemenata:

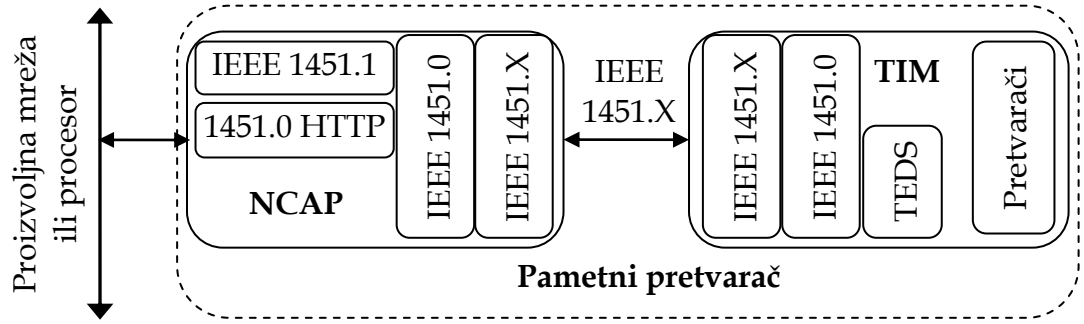
- Analizirana je upotreba univerzalnog mrežnog interfejsa za pristup različitim mrežnim entitetima, poput pametnih pretvarača, algoritama za obradu podataka i ulazno-izlaznih uređaja.
- Uveden je model virtuelnog pametnog pretvarača, koji se može smatrati formom za integraciju mrežnih entiteta koji nisu konkretni pametni pretvarači.
- Razvijen je model mreže sa razmenom podataka između proizvoljnih mrežnih entiteta posredstvom centralnog servera, koji istovremeno ima i ulogu menadžera za konfiguraciju mreže.
- Razvijen je model direktnog prenosa podataka između proizvoljnih mrežnih entiteta, sa rasterećenjem centralnog servera koji zadržava funkciju menadžera mreže.
- Uvedena je podrška za implementaciju servisno orijentisanih mreža pametnih pretvarača sa karakteristikama sistema za rad u realnom vremenu.

2. PREGLED STANJA U NAUČNOJ OBLASTI

Po pitanju standardizacije, u oblasti se posebno ističu specifikacije IEEE 1451 familije standarda, kao i specifikacije konzorcijuma Open Geospatial Consortium (OGC), pa je u ovom poglavlju naveden pregled definisanih modela. I pored toga što novi model servisno orijentisanih mreža koji će biti opisan u ovoj tezi nije striktno vezan za IEEE 1451 specifikaciju, dati standard je korišćen kao polazna osnova, pa će detaljnije biti objašnjeni interfejsi, protokoli i specifikacije definisani IEEE 1451.0 standardom [2]. Takođe, u ovom poglavlju će biti objašnjen i već postojeći koncept nadogradnje IEEE 1451.0 modela upotrebom mrežnih servisa.

2.1. Standardizacija pametnih pretvarača i senzorskih mreža

IEEE 1451.0 standard objedinjuje zajedničke funkcije pametnih pretvarača, komunikacione protokole i uvodi definiciju elektronskih specifikacija uređaja poznatu pod nazivom Transducer Electronic Data Sheet (TEDS) [2]. Standard IEEE 1451.0, kao bazni član IEEE 1451 familije, daje novi koncept dizajna pametnih pretvarača koji ima za cilj da olakša implementaciju pojedinačnih uređaja i omogući interoperabilnost između uređaja različitih proizvođača. Sam koncept pametnog pretvarača u okviru standarda podrazumeva da se uređaj sastoji iz dva dela: mrežni procesor - Network Capable Application Processor (NCAP) i interfejsni modul pretvarača - Transducer Interface Module (TIM), čiji je uprošćeni prikaz dat slikom 2.1. Interfejsni modul pretvarača obezbeđuje pristup TEDS sadržaju i samim pretvaračima, dok mrežni procesor obezbeđuje izlaz pametnog pretvarača na mrežu ili njegovo povezivanje sa procesorom za dalju obradu podataka. Implementacija mrežnog procesora i interfejsnog modula pametnog pretvarača je olakšana uvođenjem slojevite arhitekture, tako da su standardne funkcije pametnog pretvarača grupisane u servisni interfejs IEEE 1451.0 sloja, dok su komunikacione funkcije izolovane u IEEE 1451.X sloj.



Slika 2.1. Koncept mrežnog procesora i interfejsnog modula pretvarača prema IEEE 1451.0 standardu.

Povezivanje NCAP i TIM modula moguće je ostvariti prema definicijama standarda označenih na slici sa IEEE 1451.X [3-6], pri čemu važi sledeće:

- X=2 označava standard IEEE 1451.2, za digitalno žično povezivanje od tačke do tačke (point-to-point veza), pri čemu se u osnovnoj postavci standarda koristi posebno definisani interfejs Transducer Independent Interface (TII), ili neki od često korišćenih serijskih interfejsa poput USB, RS-232, SPI, I2C;
- X=3 označava standard kojim se definiše digitalno povezivanje mrežnog procesora i pretvaračkog modula na zajedničku magistralu (multi-drop veza), tako da su niži slojevi komunikacionog steka bazirani na HomePNA standardu;
- X=5 označava standard koji definiše bežičnu vezu TIM i NCAP modula putem uobičajenih bežičnih interfejsa kao što su 802.11 (WiFi), 802.15.1 (Bluetooth), 802.15.4 (ZigBee) i drugi, pri čemu je dozvoljena i međusobna komunikacija bežičnih TIM modula;
- X=6 označava IEEE 1451.6 standard za povezivanje mrežnog procesora i pretvaračkog modula upotrebom CANopen interfejsa;
- X=7 daje varijantu IEEE 1451.7 standarda kojim su definisani formati podataka za potrebe komunikacije sistema na bazi Radio Frequency Identification (RFID) tehnologije, sa pametnim RFID tagovima koji obuhvataju senzore i aktuatore.

Pored navedenog, standardom IEEE 1451.1 [7] definiše se objektni model mrežnog procesora koji obuhvata mrežno neutralni interfejs za povezivanje procesora sa mrežom, senzorima i aktuatorima. Objektni model se može koristiti kao osnova za implementaciju aplikacionog koda za izvršavanje na NCAP procesoru. Međutim, korišćenje IEEE 1451.1 modela nije obavezno u

slučaju realizacije IEEE 1451.0 uređaja, već je to izbor koji se pravi u vreme dizajna. Takođe, poseban slučaj predstavlja standard IEEE 1451.4 [8] koji definiše mešoviti interfejs za prenos analognih signala pretvarača i digitalnih TEDS podataka. Sa obzirom na to da su IEEE 1451.4 TEDS komponente posebno definisane, tako da se omogući njihovo uvođenje u klasične analogne pretvarače, prilikom izrade IEEE 1451.0-kompatibilnog uređaja potrebno je obezbediti odgovarajuću konverziju između datih TEDS i 1451.0 TEDS.

U osnovnoj postavci NCAP upravlja radom TIM modula slanjem standardnih ili posebno definisanih komandi preko IEEE 1451.X komunikacionog sloja. U tom slučaju, formati poruka koje razmenjuju NCAP i TIM putem različitih fizičkih interfejsa su standardizovani i u takvim formatima se mogu prosleđivati komandni zahtevi i odgovori. Pritom, IEEE 1451.0 sloj koristi uslužnu biblioteku 1451.X sloja za kodovanje željene komande u standardizovani format. Dobijena kodovana komanda se potom od strane IEEE 1451.0 sloja vraća 1451.X sloju u vidu niza bajtova, spremnog za dalju obradu i prenos putem odgovarajućeg 1451.X fizičkog medijuma.

IEEE 1451.X sloju se pristupa putem skupa API funkcija dostupnih programeru koji želi da realizuje logiku pretvarača na nivou IEEE 1451.0 sloja. Time se postiže da prilikom programiranja IEEE 1451.0 sloja budu sakriveni detalji implementacije NCAP-TIM komunikacije. Takođe, aplikacija koja se izvršava na NCAP-u poziva funkcije IEEE 1451.0 sloja date kroz API definicije, pri čemu standard ne postavlja ograničenja po pitanju detalja njihove implementacije. Tipično, u okviru IEEE 1451.0 funkcije (servisa) na NCAP strani izvršava se kodovanje komandi za slanje TIM modulu, koji primljene komande izvršava pozivanjem servisa lokalnog IEEE 1451.0 sloja, a zatim se po potrebi rezultat izvršavanja komande koduje u poruku odgovora. Iako 1451.0 servisi na NCAP i TIM strani nisu identični, mnogi 1451.0 servisi NCAP-a imaju svoje parnjake u okviru 1451.0 sloja na TIM strani.

Pri tome, komunikacioni 1451.X interfejs i njegova interakcija sa 1451.0 slojem je osmišljena tako da omogući plug & play povezivanje TIM modula bez

upotrebe posebnih drajvera i profila uređaja. Novopriključeni TIM modul u svojoj memoriji sadrži meta podatke u standardnom TEDS formatu, dostupne NCAP-u, tako da je moguće ostvariti identifikaciju i upravljanje pojedinačnim pretvaračima i celokupni modulom. Stanje pojedinačnih pretvarača, kao i stanje celokupnog TIM modula, dobija se očitavanjem statusnih registara preko posebnih komandi koje se šalju TIM modulu. Standardom je data i mogućnost generisanja servisnih zahteva od strane TIM-a, čime se izbegava poliranje TIM registara od strane NCAP procesora. Na strani aplikacije, za pristup 1451.0 servisima preko mreže, definisan je *Hypertext Transfer Protocol* (HTTP) 1451.0 API koji omogućava pozivanje standardnih IEEE 1451.0 operacija NCAP-a putem HTTP protokola.

Pored upotrebe IEEE 1451 familije standarda, u oblasti se ističe i aktivnost OGC konzorcijuma, koji se bavi donošenjem standarda u cilju poboljšanja interoperabilnosti uređaja za razmenu lokacijskih informacija i zasniva se na saradnji različitih strana poput univerziteta i industrije. U okviru inicijative OGC - Sensor Web Enablement (OGC-SWE), dat je set specifikacija koje obezbeđuju interoperabilnost i razmenu podataka i meta podataka u senzorskim mrežama, kao geografski distribuiranim sistemima [9-12]. Definisane su sledeće specifikacije:

- O&M (*Observations & Measurements*) definiše modele i XML šeme za kodovanje postupka opservacije senzora i rezultata opservacije. Koristi se termin opservacija koji ima šire značenje od samog merenja. Procena vrednosti nekog svojstva putem algoritma jeste primer opservacije koja nije merenje.
- SensorML (*Sensor Model Language*) definiše modele i XML šeme za opis procesa merenja i procesa obrade mernih podataka. Prema tome, sve komponente sistema, a samim tim i pretvarači, se smatraju procesima koji se mogu međusobno povezivati.
- TML (*Transducer Markup Language*) definiše konceptualni model i XML šeme za opis pretvarača i daje podršku za uspostavljanje tokova podataka prema senzorskim sistemima u realnom vremenu i prema arhivama podataka.
- SOS (*Sensor Observations Service*) definiše interfejs Web servisa za upravljanje i pristup meta podacima i podacima opservacija

sistema koji obuhvata različite senzore (seizmički senzori, podaci sa satelita i dr.). Za opis opservacija se koristi O&M format, a za opis senzora SensorML.

- SPS (*Sensor Planning Service*) definiše interfejs za pristup informacijama koje se tiču mogućnosti određenog entiteta (npr. senzora ili simulacije), opcija parametrizacije, kao i interfejsa za pokretanje i kontrolu taskova na strani tog entiteta. Sistem se u ovom slučaju posmatra kao *black box* objekat sa kojim se interaguje posredstvom taskova i parametara, za razliku od slučaja direktnog pristupa podacima upotrebom SOS modela.
- SAS (*Sensor Alert Service*) definiše interfejs za slanje upozorenja senzorskih čvorova u publish/subscribe modelu. Slanje obične merne vrednosti od strane senzorskog čvora se takođe registruje kao upozorenje u sistemu, pri čemu prijemni SAS čvor vrši dalju analizu merne vrednosti i generiše novi vid upozorenja ukoliko je to potrebno.
- WNS (*Web Notification Services*) definiše interfejs Web servisa za slanje obaveštenja opšte namene. Komunikacija sa registrovanim korisnikom se odvija sinhrono, kada se od klijenta koji prima obaveštenje ne očekuje odgovor, ili asinhrono, kada se od klijenta očekuje poruka odgovora. Model se efikasno kombinuje sa SPS servisima za startovanje taskova, kako bi se registrovani korisnik obavestio o statusu izvršenja taska (npr. obaveštenje o završenoj akviziciji podataka).

Slično kao u slučaju IEEE 1451 familije standarda čije su zajedničke karakteristike objedinjene IEEE 1451.0 specifikacijom, u okviru OGC-SWE seta se javljaju specifikacije *SWE Common Data Model* i *SWE Service Model*, o čijoj upotrebi je diskutovano u radu [12]. Model podataka dat prvom specifikacijom služi za opis opservacija senzora na uobičajen način, tako da dekapluje same rezultate opservacija i meta podatke, dok je drugom specifikacijom dat model servisa koji objedinjuje zajedničke tipove i interfejse koji se javljaju kroz specifikacije servisa (npr. SOS i SPS). Objedinjavanjem zajedničkih karakteristika kroz posebne specifikacije, dobija se očuvanje funkcionalnosti i povezivanje ostalih specifikacija iz date grupe u jednu funkcionalnu celinu. Takođe, SAS model definiše sopstveni skup pravila neoslanjajući se na postojeće IT standarde, pa je taj problem rešen uvođenjem novih specifikacija poput *Sensor Event Service* (SES) i *OGC Event Service*.

Opisani standardi Web servisa za pristup arhivama senzorskih podataka, pirstup senzorima u realnom vremenu, slanje obaveštenja u alarmnim situacijama, zajedno sa pratećom dokumentacijom standardnih modela senzora i procesa merenja, mogu se koristiti za formiranje globalnih mreža u kojima se razmenjuju geolokacijske informacije. Time je dat komplementaran skup specifikacija u odnosu na specifikaciju IEEE 1451.0 standarda koja podrazumeva realizaciju funkcionalnosti na nivou samog uređaja. Nasuprot tome, OGC-SWE daje rešenje za podatke koji su već dopremljeni do nivoa mreže i koji se koriste u Web okruženju.

2.2. Pametni pretvarači prema IEEE 1451.0 standardu

Standard IEEE 1451.0 definiše upotrebu lokalnih servisa, svrstanih u IEEE 1451.0 sloj pametnog pretvarača prema modelu koji je dat slikom 2.1. Takvi lokalni servisi su dostupni NCAP aplikacionom sloju, sadrže skup uobičajenih operacija pametnih pretvarača i apstrakuju detalje specifičnih implementacija. U nastavku će biti dat detaljniji opis servisnog interfejsa, modela podataka, elektronskih specifikacija, kao i HTTP interfejsa za pristup pametnom pretvaraču.

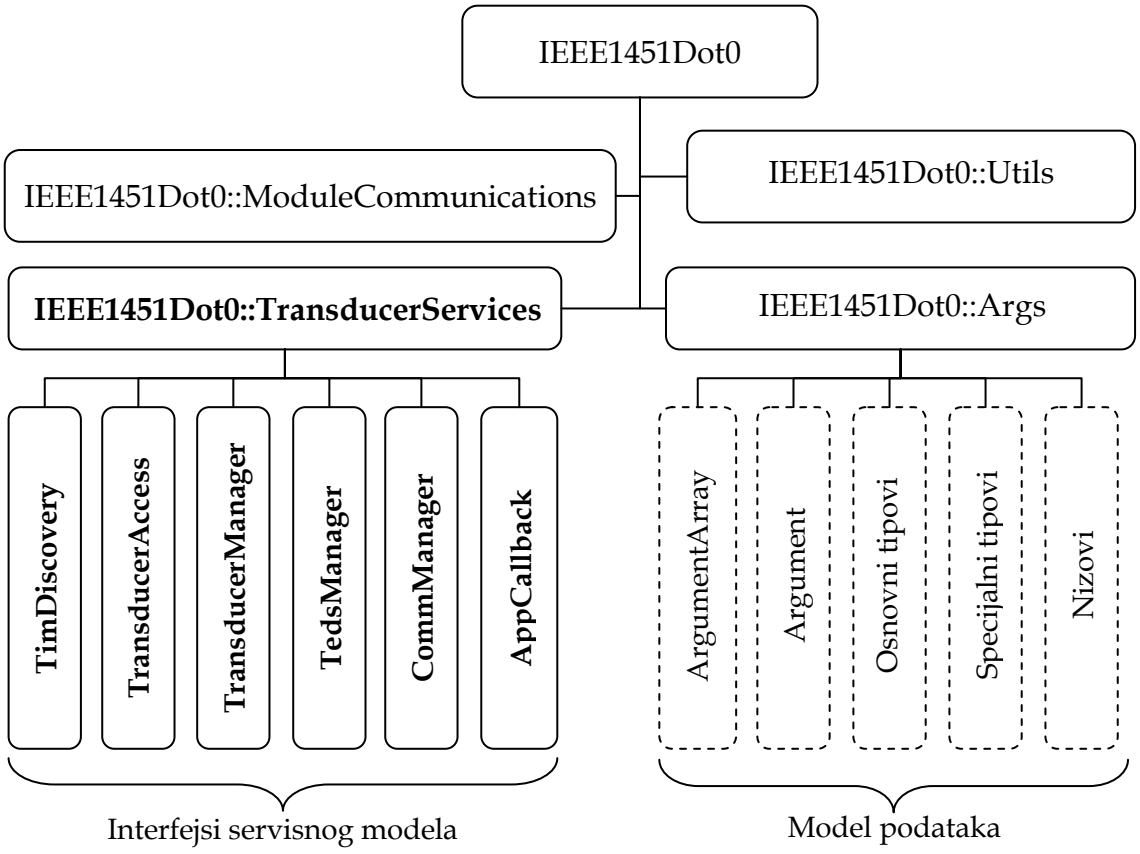
Izuzimajući specifikaciju HTTP API-ja, funkcije definisane standardom se grupišu u dva glavna interfejsa koja međusobno povezuju tri sloja na strani NCAP procesora: aplikacioni, IEEE 1451.0 i IEEE 1451.X sloj. Servisni interfejs pretvarača (Transducer Services Interface) podrazumeva API koji koriste merno-kontrolne aplikacije NCAP-a za pristup IEEE 1451.0 sloju prikazanom na slici 2.1, dok je komunikacioni interfejs modula (Module Communication Interface) dat API definicijom za povezivanje IEEE 1451.0 sloja sa komunikacionim slojem IEEE 1451.X. Prvi interfejs pretvarača, odnosno njegov servisni API, koriste aplikacije NCAP-a za upis/očitavanje pretvaračkih kanala, upis/čitanje TEDS sadržaja, kao i za slanje konfiguracionih, kontrolnih i operacionih komandi ka TIM modulu. Drugi interfejs, odnosno komunikacioni interfejs pretvarača, se implementira simetrično, na NCAP i na TIM strani, i obuhvata metode IEEE 1451.X sloja koje poziva IEEE 1451.0, ali i metode koje su

svrstane u IEEE 1451.0 sloj i poziva ih IEEE 1451.X za potrebe isporuke sadržaja koji se prenosi.

Kako bi definicija navedenih interfejsa prilikom implementacije bila nezavisna od izbora programskog jezika koristi se IDL (Interface Definition Language) prema ISO 14750 specifikaciji, za opis funkcija i njihovih parametara i rezultata. Specifikacijom se definišu četiri modula koja pripadaju modulu najvišeg nivoa *IEEE1451Dot0* prema slici 2.2. IEEE 1451.0 servisi pametnog pretvarača su svrstani u modul *TransducerServices*, a metode komunikacionog sloja u modul *ModuleCommunications*. Takođe, definišu se *Utils* uslužne metode za prevođenje podataka IEEE 1451.0 sloja u format poruka prihvatljivih za prenos putem IEEE 1451.X. U okviru modula *Args* su date IDL definicije osnovnih, nizovnih, kao i specijalnih tipova neophodnih za realizaciju pametnog pretvarača, poput tipa za skladištenje jedinstvenog identifikacionog broja TIM modula, mernih jedinica i tipa za rad sa vremenskom referencom.

Prema tome, može se reći da modul *TransducerServices* predstavlja opis servisnog modela, dok je model podataka opisan pomoću *Args*. Metode obuhvaćene servisnim modelom su prema nameni svrstane u različite IDL interfejse: *TimDiscovery*, *TransducerAccess*, *TransducerManager*, *TedsManager*, *CommManager* i *AppCallback*, čiji je opis dat na slici. U tipičnom primeru, aplikacija mrežnog procesora koristi *TimDiscovery* metode za otkrivanje dostupnih TIM-ova i kanala, *TedsManager* za učitavanje njihovih TEDS sadržaja i *TransducerAccess* za očitavanje mernih/kontrolnih podataka.

Međutim, tip podataka koji se dobija od strane priključenog TIM-a ne mora uvek biti isti i zavisi od njegove konkretne implementacije. Tako se očitavanjem mernih vrednosti senzora može dobiti celobrojni podatak, niz karaktera ili podatak nekog drugog tipa. Kako bi se olakšao prenos i upotreba podataka na IEEE 1451.0 sloju i višim slojevima aplikacija, modelom podataka su definisane i IDL strukture. U okviru određene implementacije, date strukture se mapiraju prema strukturama izabranog programskog jezika: C strukture, C++/Java klase i slično.



TimDiscovery – Metode dostupne aplikacijama za otkrivanje komunikacionih modula, TIM-ova i pretvaračkih kanala su grupisane u ovaj interfejs.
TransducerAccess – Metode koje koristi aplikacija mrežnog procesora NCAP za pristup kanalima senzora i aktuatora.
TransducerManager – Metode ovog interfejsa koristi aplikacija kojoj je potreban veći nivo kontrole nad TIM modulom (npr. zaključavanje TIM-a i dobijanje ekskluzivnog pristupa ili slanje proizvoljnih komandi).
TedsManager – Metode za čitanje i upis TEDS sadržaja. Ova klasa ima nadležnost nad TEDS sadržajem koji je keširan na strani NCAP-a.
CommManager – Metode za pristup komunikacionom modulu lokalnog uređaja.
AppCallback – Za napredne aplikacije koje registruju callback funkcije.

Slika 2.2. Prikaz organizacije modula i interfejsa datih API definicijom IEEE 1451.0 standarda.

Korišćenjem IDL definicije, uvodi se struktura generičkog sadržaoca *Argument*, koji ima mogućnost čuvanja podataka bilo kog tipa iz skupa prostih, nizovnih ili specijalnih tipova. Kao takav, *Argument* se u konkretnom

programskom jeziku može implementirati u vidu diskriminišuće unije ili klase/strukture koja sadrži polja svih mogućih tipova, tako da se u jednom trenutku koristi samo jedno polje. Tipski kod koji je dat u vidu enumeracije je uvek prisutan i jednoznačno određuje tip podatka koji je trenutno sadržan u argumentu. Takođe, modelom podataka se predviđa i upotreba interfejsa *ArgumentArray* u koji su grupisane metode za rad sa nizovima argumenata, odnosno podacima tipa *Argument*.

Primeri IDL definicija prostih i nizovnih tipova koji se mogu prenosti pomoću *Argument* strukture, dati su tabelom 2.1. Primerom u prvoj koloni su date definicije prostog tipa i njemu pridruženog niza, dok je drugom kolonom dat opis osnovnog tipa. Pri tome, bajtovi se prema IEEE 1451.0 specifikaciji preciznije nazivaju oktetima.

Generička struktura podataka *ArgumentArray* jeste najčešće korišćeni tip podataka u okviru IEEE 1451.0 i aplikacionog sloja. *Argument* kao član niza *ArgumentArray* je jako fleksibilna struktura koja omogućava prenos različitih tipova podataka kroz sistem, bez poznavanja određenog tipa u vremenu prevođenja izvornog koda programa. Na taj način je moguće definisati univerzalne metode servisnog interfejsa tako da pokrivaju različite slučajeve kodovanja podataka u argument. Na slici 2.3 (a) je prikazana IDL definicija metode *readData* koja pripada interfejsu *TransducerAccess* i koristi se za pristup mernim podacima senzora, sa izlaznim parametrom *result* tipa generičkog niza *ArgumentArray*. Ulazni parametar *transCommId* označava broj komunikacione sesije za određeni pretvarački kanal i dat je osnovnim celobrojnim tipom *Uint16*. Tip *TimeDuration* predstavlja specijalni tip standarda i u ovom slučaju se koristi za specificiranje vremenskog perioda (ulazni parametar *timeout*) koju koristi *timeout* logika IEEE 1451.0 sloja za kontrolu komunikacije.

Parametar *SamplingMode* služi za izbor *trigger* režima senzora koji se može menjati tokom višestrukih poziva metode *readData*. Povratna vrednost metode je *Uint16* tipa i predstavlja kod greške, pri čemu vrednost 0 podrazumeva ispravno izvršavanje funkcije. Pomoću 3 bita najveće težine se

koduje sloj na kome je došlo do greške (lokalni ili udaljeni IEEE 1451.0/IEEE 1451.X, ili udaljeni aplikacioni sloj), dok se sa nižih 13 bita koduje sama greška u vidu neoznačenog celobrojnog podatka.

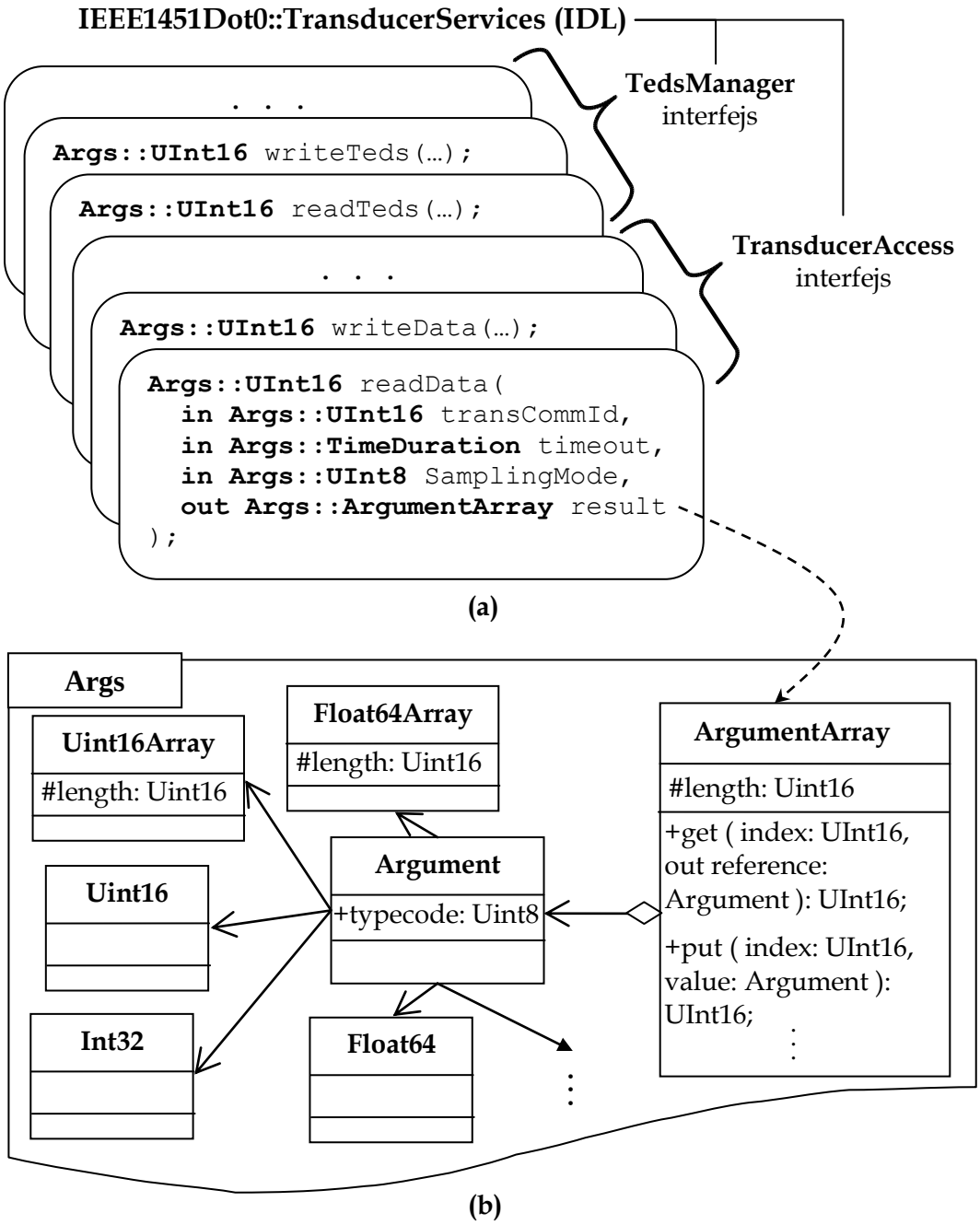
Tabela 2.1. Primeri IDL definicija osnovnih tipova i nizova osnovnih tipova prema IEEE 1451.0. Nizovni tipovi sadrže same podatke, kao i podatak od dužini sekvence tipa Uint16.

IDL PRIMER: OSNOVNI TIP I IZVEDENI NIZ	OPIS
`typedef octet UInt16;` `typedef sequence<UInt16> UInt16Array;`	`UInt16`: Neoznačena celobrojna vrednost od 0 do 65 535 (2 okteta).
`typedef long Int32;` `typedef sequence<Int32> Int32Array;`	`Int32`: Označena celobrojna vrednost od –2 147 483 648 do 2 147 483 647 (4 okteta).
`typedef double Float64;` `typedef sequence<Float64> Float64Array;`	`Float64`: Dvostruka preciznost u sistemu sa pokretnom tačkom prema IEEE std. 754-1985 (8 okteta).

Sama implementacija metode *readData* podrazumeva višestruko slanje komandi ka TIM modulu za očitavanje konačnog rezultata procesa odabiranja i TEDS podataka neophodnih za kodovanje argumenata generičkog niza. U slučaju da su obezbeđeni korekcioni koeficijenti putem TEDS-a pridruženog mernom kanalu, tipično je obavljanje korekcije u okviru same metode *readData*.

Tada se korekcioni koeficijenti primenjuju na rezultat očitavanja pretvaračkog kanala i rezultat korekcije, koji je tipično *Float32* tipa, smešta se u argument sa odgovarajućim tipskim kodom. Ukoliko pretvarački kanal ne obezbeđuje korekcione koeficijente putem TEDS-a, argumentu će biti pridružena vrednost u izvornom tipu pretvaračkog kanala.

Detalji implementacije metode *readData* nisu dati standardom i zavise isključivo od odluka dizajnera. I pored toga NCAP aplikacija očitava pretvarački kanal uvek na isti način definisan servisnim API-jem. Ilustracija odnosa generičkog kontejnera podataka i drugih tipova standarda data je slikom 2.3 (b). Po sličnom principu su date IDL definicije drugih metoda servisnog interfejsa, čiji su detalji dati standardom [2].



Slika 2.3. Upotreba tipova definisanih *Args* modulom u definiciji servisa: (a) Parametri metode (servisa) i povratna vrednost na bazi *Args* tipova; (b) Deo *Unified Modeling Language* (UML) dijagrama koji opisuje *ArgumentArray*, generički kontejner za prenos podataka proizvoljnog tipa.

Pored upotrebe TEDS sadržaja u metodama servisnog interfejsa, ovaj sadržaj se može proslediti i aplikacionom sloju NCAP-a korišćenjem metoda *TedsManager* interfejsa. Pritom postoje dve varijante prosleđivanja sadržaja:

„sirovi“ TEDS podaci se dobijaju kao niz bajtova, u binarnom formatu definisanom standardom; ili se binarna polja TEDS-a dekoduju u metodi *readTeds* servisnog sloja, a zatim prosleđuju dalje u vidu argumenata generičkog niza. Druga varijanta omogućava NCAP aplikaciji prijem već dekodovanog TEDS-a, kome pristupa putem *ArgumentArray* interfejsa, gde je svaki pojedinačni atribut TEDS-a kodovan u jedan argument. Pretraga argumenata se u tom slučaju izvodi po imenu atributa datog standardom ili po indeksu argumenta. Neki od često korišćenih TEDS-ova su opisani tabelom 2.2.

TEDS komponente predstavljaju važnu stavku u ostvarivanju automatskog povezivanja i korišćenja pametnog pretvarača, sa obzirom na to da se mogu očitavati mašinskim putem bez intervencije korisnika. TEDS komponente su posebno važne, jer preko unapred definisanih interfejsa pružaju podatke o samom pretvaračkom modulu, koji su razumljivi operateru (u vidu tekstualnih TEDS), kao i podatke koji su razumljivi računaru (binarni TEDS).

Bez obzira na vrstu, svaki TEDS blok je podeljen na više polja koja se implementiraju u formatu tip/dužina/vrednost. „Tip“ je dužine jednog okteta i služi za identifikaciju polja, što je funkcija slična onoj koju imaju tagovi u XML slučaju. „Dužina“ označava broj okteta koji se nalaze u polju „vrednost“, pri čemu „vrednost“ sadrži same TEDS podatke. Kada su u pitanju tekstualni TEDS-ovi, polja u formi uređenih torki (tip, dužina, vrednost) se koriste za kreiranje strukture direktorijuma, kako bi se omogućio pristup različitim sekcijama tekstualne porcije TEDS-a. Osim jednooktetnog koda koji označava tip i koji se nalazi na početku svakog TEDS polja, standardom se dodeljuje i ime svakom od datih polja. Prema tome, TEDS polja se mogu prenositi putem *ArgumentArray* interfejsa koji omogućava pretragu polja po imenu.

U slučaju prenosa TEDS podataka treba primetiti da se dekodovanje primljenog sadržaja izvršava u dva koraka na NCAP strani. U prvom koraku se dekoduje sama poruka dobijena od strane TIM-a koja sadrži TEDS, upotrebom uslužnih metoda modula *Utils* koje poziva IEEE 1451.0 sloj. U drugom koraku se dekoduje dobijeni TEDS u okviru metode IEEE 1451.0 sloja ili se, ipak,

prosleđuje višim slojevima koji imaju mogućnost TEDS dekodovanja. Rezultat prvog dekodovanja je generički niz (*ArgumentArray)* koji čitav TEDS ili pak njegov segment sadrži u samo jednom argumentu, u vidu niza bajtova.

Tabela 2.2. Opis često korišćenih TEDS struktura datih IEEE 1451.0 standardom.

NAZIV	TEDS KOD	OPIS
Meta-TEDS *(obavezni)*	1	Specificira vrednosti operacionih vremenskih parametara TIM-a koji se dobijaju u najgorem slučaju, kao i relacije između postojećih kanala. NCAP koristi ove vremenske parametre za podešavanje timeout perioda komunikacije.
TransducerChannel TEDS *(obavezni)*	3	Ovaj TEDS blok daje detaljne informacije o pretvaraču. Navodi se fizički parametar koji se meri ili kontroliše, radni opseg pretvaračkog kanala, karakteristike digitalnog ulaza/izlaza, režimi rada i vremenski parametri.
User's Transducer Name TEDS *(obavezni)*	12	Struktura u koju korisnik pretvarača upisuje ime, prema kome će pretvarač biti prepoznat u okviru sistema.
PHY TEDS *(obavezni)*	13	Sadržaj ovog bloka zavisi od fizičkog komunikacionog medijuma koji se koristi za povezivanje TIM-a sa NCAP procesorom. Format nije definisan standardom IEEE 1451.0, mada je definisan metod za pristup TEDS-u.
Calibration TEDS *(opcioni)*	5	U okviru ovog TEDS-a se nalaze sve informacije potrebne softveru za korekciju, koje se odnose na određeni pretvarački kanal. U zavisnosti od vrednosti kalibracionog ključa koji je deo TEDS-a, korekcija se sprovodi na TIM-u ili NCAP-u (ukoliko funkcija nije dislocirana sa NCAP-a na neki drugi mrežni čvor).
Text-based TEDS *(opcioni)*	/	Funkcija ovih TEDS-ova je obezbeđivanje informacija za prikaz operateru pomoću displeja. Struktura se sastoji iz jednog ili više blokova koji sadrže tekstualne informacije (svaki na posebnom jeziku). Pristupni TEDS kod zavisi od vrste tekstualnog TEDS-a, pri čemu je standardom definisano 6 različitih vrsta. Formati predloženi standardom nisu obavezni.

Osim TEDS sadržaja, rezultujući generički niz sadrži i dodatni argument koji daje ofset početka TEDS segmenta, čime se omogućava parcijalno čitanje dugačkih TEDS-ova, koji prevazilaze smeštajni kapacitet TIM poruke. Nakon učitavanja čitavog TEDS-a, *readTeds* metoda IEEE 1451.0 sloja analizira niz TEDS bajtova, prepoznajući pojedinačna binarne torke u formatu (tip, dužina, vrednost). Pojedinačna TEDS polja se zatim smeštaju u zasebne argumente novoformiranog generičkog niz tipa *ArgumentArray*, koji se zatim kao rezultat prosleđuje aplikaciji NCAP-a. Međutim, ukoliko je dostupan u kešu, TEDS

sadržaj se u slučaju date metode učitava iz lokalne NCAP memorije. U okviru interfejsa je data i IDL definicija metode *readRawTeds* koja uvek učitava TEDS sadržaj iz TIM-a zaobilazeći keš, i prosleđuje ga aplikaciji u vidu niza bajtova. Tada se drugi stepen dekodovanja izvršava u okviru NCAP aplikacije ili , ipak, aplikacija vrši prosleđivanje sadržaja dislociranom mrežnom čvoru sa mogućnošću dekodovanja.

Za potrebe mrežnog pristupa IEEE 1451.0 servisima predviđena je upotreba adaptera na bazi IEEE 1451.1 standarda ili se koristi IEEE 1451.0 HTTP API. U okviru druge opcije se omogućava pristup IEEE 1451.0 sloju putem HTTP GET i HTTP PUT metoda. Tabelom 2.3 je prikazan format URL-a kojim se poziva odgovarajuća metoda IEEE 1451.0 servisnog API-ja, prema specifikaciji RFC 2616, odnosno prema sledećem formatu: http://<host>:<port>/<path>?<parameters>. Primerom je data definicija polja <path> za slučaj pozivanja readData metode, dok se na sličan način definiše putanja za poziv ostalih metoda servisnog interfejsa pametnog pretvarača. Poruka zahteva se u datom formatu šalje HTTP serveru koji se izvršava na NCAP procesoru, nakon čega server procesira zahtev i poziva odgovarajući IEEE 1451.0 servis. Rezultat izvršavanja IEEE 1451.0 operacije se vraća klijentu, kao HTTP odgvor servera.

Prema datoj specifikaciji HTTP klijent specificira format odgovora koji može biti u tekstualnom, HTML ili XML formatu. Za slučaj XML odgovora, standardom su propisani formati u XML Schema jeziku. Ovoga puta, ArgumentArray kao generički niz podataka je umesto u IDL-u definisan kao kompleksan XML tip u XML Schema jeziku. Prosti tipovi IDL-a su definisani kao prosti tipovi XML Schema jezika, kao bi se dobila osnova za formiranje složenih tipova.

Primenom tehnologije Web servisa uvodi se veći stepen automatizacije u proces razmene poruka, nego u slučaju korišćenja samog HTTP protokola koji zahteva dodatni programerski napor za serijalizaciju/deserijalizaciju XML sadržaja. Striktno govoreći, mrežni servisi nisu predloženi u okviru standarda,

ali jesu bazirani na IEEE 1451.0 specifikaciji. U nastavku će biti analizirani modeli implementacije koji se zasnivaju na korišćenju ovakvih mrežnih servisa.

Tabela 2.3. Format poruke namenjen HTTP komunikaciji.

POLJE	DEFINICIJA	PRIMER
<host>	Host deo specificira ciljano ime domena IEEE 1451 mrežnog čvora.	<*host*> = "192.168.1.101"
<port>	Opciono polje sa podrazumevanom vrednošću 80. Opcija može biti korisna za izbor posbenog porta za IEEE 1451 aplikaciju, ali može zahtevati eksplicitno podešavanje rutera kako bi se omogućila komunikacija putem nestandardnog porta.	<*port*> = "80"
<path>	Putanja IEEE 1451 servisa, uključujući i ime komande.	<*path*>="1451/TransducerAccess/ReadData"
<parameters>	Navođenje parametara komande.	<*parameters*>="timId=1&channelId=2&timeout=14&samplingMode=continuous&format=text"

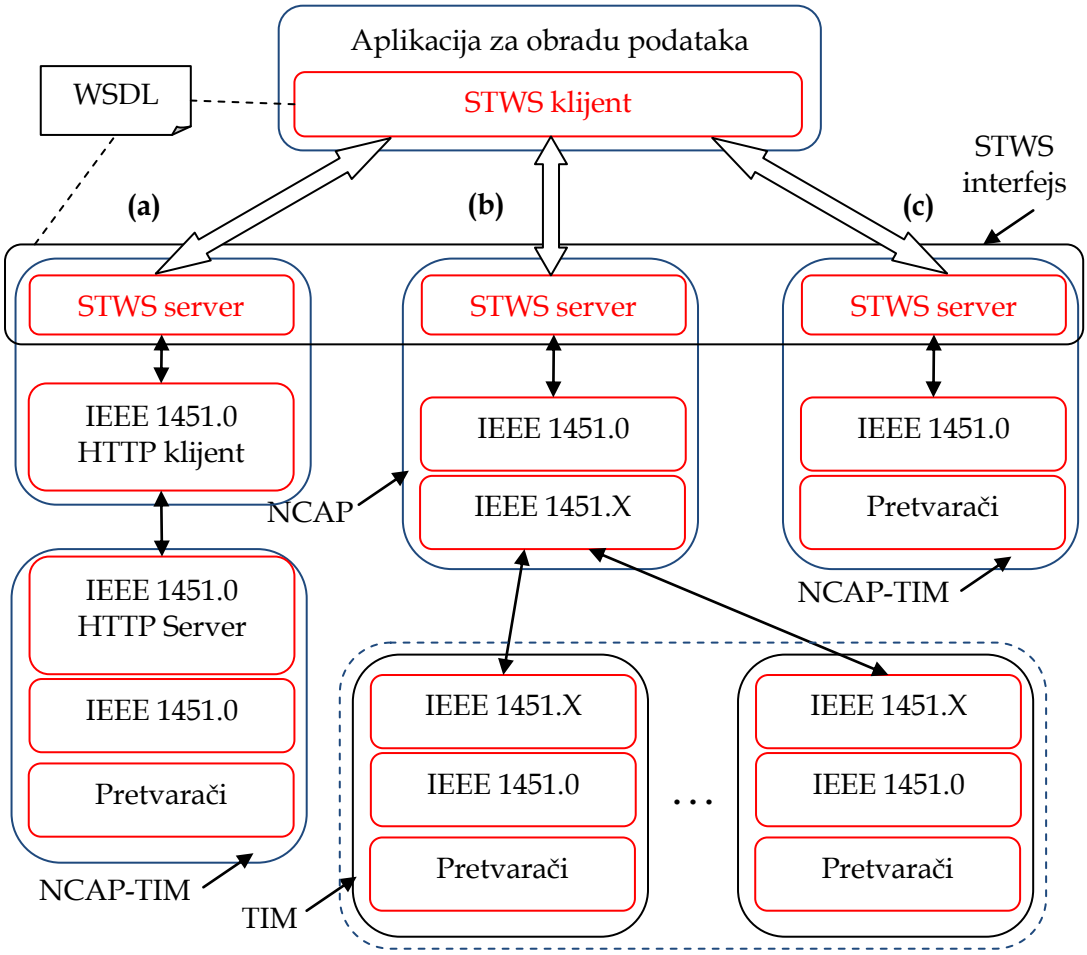
2.3. Servisno orijentisana arhitektura u mreži pametnih pretvarača

U cilju postizanja veće interoperabilnosti u mreži pametnih pretvarača, predložena je primena SOA u mreži IEEE 1451.0 uređaja [13]. Iako se SOA može realizovati na više specifičnih načina [14-18], prema potrebama aplikacija, nameće se upotreba Web servisa za pristup servisnim operacijama pametnog pretvarača kao rešenje koje nije u konfliktu sa mrežnim *firewall* i *proxy* komponentama. Takvom nadogradnjom IEEE 1451.0 standarda, dolazi se do novog koncepta u korišćenju Web servisa, u ovom slučaju za potrebe komunikacije sa pametnim pretvaračima: Smart Transducer Web Services (STWS). STWS server se tipično implementira na platformi opšte namene kao zaseban čvor ili kao deo NCAP mrežnog procesora. U slučaju da se STWS server realizuje kao zaseban mrežni čvor, potrebno je koristiti IEEE 1451.0 HTTP API za komunikaciju sa HTTP serverom na NCAP strani. U slučaju da se STWS server integriše u NCAP, nije neophodna upotreba IEEE 1451.0 HTTP API-ja. Bez obzira na izabranu realizaciju, krajnji klijent na isti način pristupa

STWS serveru koji je zaduženi za dalju komunikaciju sa pametnim pretvaračem [13].

Opšta STWS arhitektura je prikazana slikom 2.4. U slučaju (a) je prikazan scenario u kome se STWS server implementira kao zaseban čvor koji komunicira sa NCAP mrežnim procesorom putem HTTP API-ja definisanog IEEE 1451.0 standardom. Sa obzirom na to da IEEE 1451.X interfejs nije jasno vidljiv, može se smatrati da se radi o NCAP-TIM modulu koji kombinuje funkcionalnost mrežnog procesora i interfejsnog modula pretvarača. Sa stanovišta STWS servera, uređaju se pristupa na isti način kao u slučaju kada je komunikacioni interfejs NCAP-TIM barijere vidljiv. Prikaz u slučaju (b) ilustruje integraciju STWS servera i NCAP mrežnog procesora u vidu mrežnog čvora sa vidljivim IEEE 1451.X interfejsom. U ovom slučaju je upotreba 1451.0 HTTP API-ja suvišna, tako da se servisnom sloju pametnog pretvarača pristupa direktno iz STWS mrežnih servisa. Na slici 2.4 (c) je prikazana puna integracija svih modula pametnog pretvarača: mrežnog procesora NCAP, interfejsnog modula pretvarača TIM i STWS mrežnog interfejsa. U ovom slučaju se eliminišu iz upotrebe i 1451.0 HTTP API i IEEE 1451.X komponente.

Definicija STWS servisa se izvodi upotrebom *Web Service Description Language* (WSDL) jezika, koji predstavlja poseban vid XML-a za specifikaciju seta apstraktnih servisa i njihovo povezivanje sa konkretnim komunikacionim protokolima. IEEE 1451.0 servisi se definišu upotrebom IDL-a, dok se mrežni STWS servisi definišu WSDL-om koji je takođe nezavistan od izbora konkretnog programskog jezika za implementaciju servisa. Tokom definisanja STWS interfejsa, potrebno je odabrati skup novih mrežnih servisa koji će biti ponuđeni STWS klijentu. Kasnije, prilikom implementacije metoda STWS servisa, vrši se mapiranje skupa postojećih IEEE 1451.0 servisa na skup novih STWS servisa. Slika 2.5 (a) prikazuje mogućnosti kombinovanja IEEE 1451.0 servisa i STWS mrežnih servisa, odnosno mapiranje jedan-na-jedan, jedan-na-više i više-na-jedan.



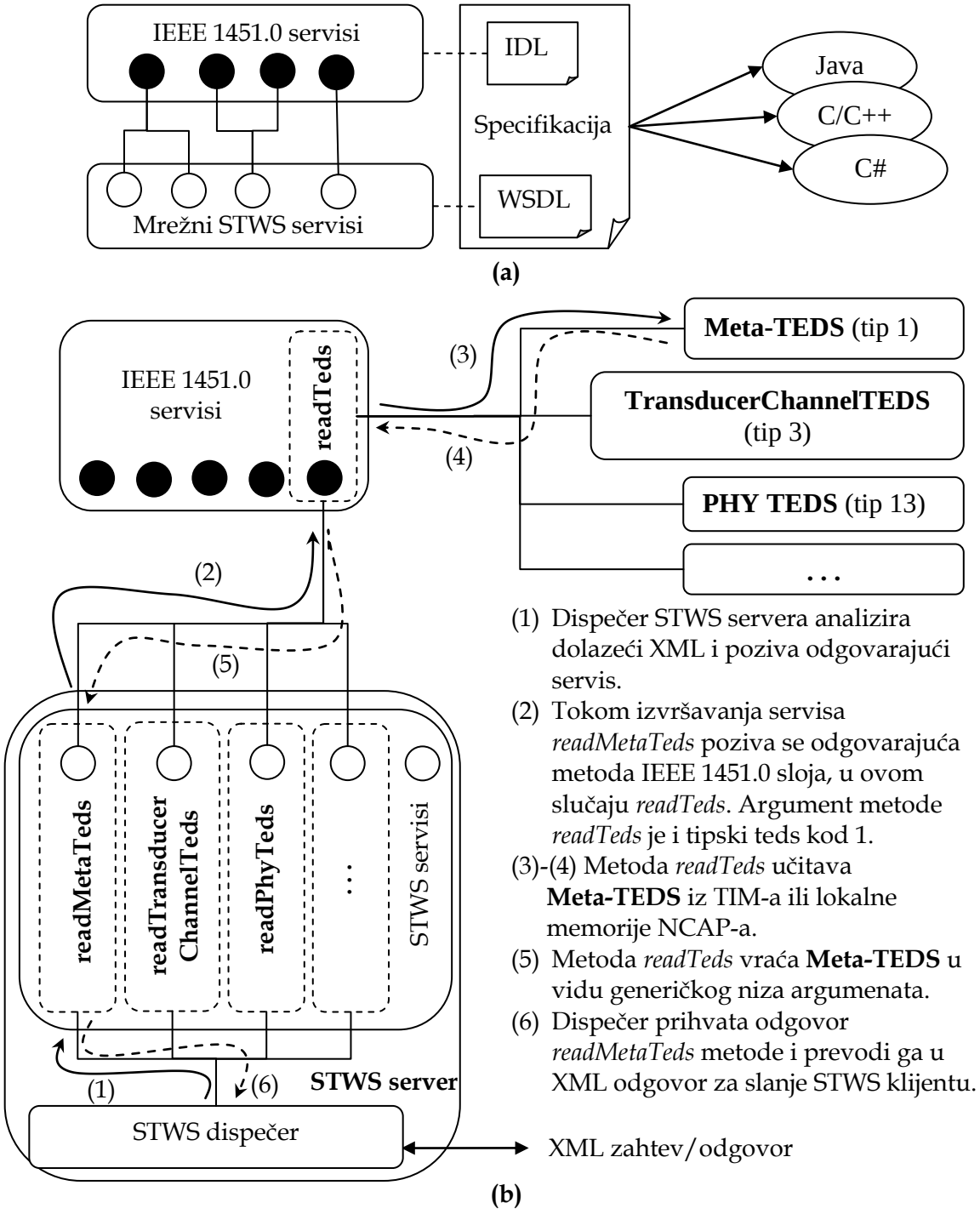
Slika 2.4. Uvođenje Web servisa u mrežu pametnih pretvarača na bazi IEEE 1451.0 standarda: (a) STWS čvor komunicira sa dislociranim NCAP-TIM modulom preko HTTP interfejsa opisanog u IEEE 1451.0; (b) NCAP modul integriše STWS server i komunicira sa TIM modulima preko IEEE 1451.X; (c) Puna integracija NCAP i TIM modula i STWS servera.

Realizacija STWS servisa prikazana u radu [19], daje primer WSDL definicije u kome se svakom od IEEE 1451.0 servisa dodeljuje jedan ili više mrežnih STWS servisa. Pogodnost uvođenja više mrežnih servisa koji se povezuju sa jednim IEEE 1451.0 servisom, ilustrovana je slikom 2.5 (b). Metoda *readTeds* u okviru IEEE 1451.0 sloja se može koristiti za očitavanje bilo kog TEDS-a, pri čemu se kao argument poziva metode koristi TEDS tipski kod. Nasuprot tome, za svaki TEDS tip se može definisati poseban STWS mrežni servis. Na taj način klijentska strana dobija informaciju o vrstama dostupnih TEDS, samo na osnovu imena STWS servisa definisanih WSDL-om. U tom

slučaju, STWS serveru se pristupa preko mrežnog klijenta implementiranog na osnovu postojećeg WSDL-a, pri čemu programer klijentske strane ne mora da vodi računa o detaljima pristupa IEEE 1451.0 sloju. Na sličan način, različitim mapiranjima, definišu se i drugi mrežni STWS servisi čime se dobija skup servisa koji je usko povezan sa specifikacijom standarda, iako tehnički nije njen deo.

XML kao često korišćeni format, pogodan je za razmenu struktuiranih podataka putem mreže u okviru distribuirane aplikacije, pa se ovde koristi za slanje zahteva STWS klijenta, odgovora STWS servera i WSDL definiciju servisa. XML "opremljen" imenskim prostorima pruža bazu za razvoj jezika poput WSDL-a i XML Schema jezika. Na početku realizacije sistema po principu odozgo na dole, potrebno je definisati mrežni interfejs koji će se kasnije koristiti za realizaciju i uniforman pristup različitm mrežnim čvorovima. Kako bi se ilustrovao proces definisanja, u nastavku će biti detaljnije analiziran WSDL sadržaj. WSDL kao specijalan slučaj korišćenja XML-a daje unapred definisanu sintaksu za definiciju Web servisa, tako da se definiciona struktura sastoji od 6 elemenata prikazanih tabelom 2.4.

WSDL tipovi poruka se definišu upotrebom XML Schema jezika, tako da pripadaju elementu *types* iz tabele 2.4. Specifikacija tipova predstavlja šablon prema kome se instanciraju XML poruke i vrši njihova validacija, tako da svaki mrežni čvor ima „očekivanja" po pitanju strukture dolazeće poruke. Mrežni čvor na osnovu XML Schema šablona, pored validacije, vrši konverziju dolazeće XML poruke u odgovarajuću strukturu korišćenog programskog jezika. Važi i obrnuto, pri čemu se podaci iz programske memorije na osnovu šablona prevode u XML za slanje ka odredišnom mrežnom čvoru. Pritom, mrežni čvorovi koji vrše razmenu ovakvih poruka (npr. server i klijent) mogu biti implementirani pomoću različitih programskih jezika.



Slika 2.5. Kombinovanje metoda IEEE 1451.0 i STWS servisa: (a) Mapiranje servisa i njihova specifikacija; (b) Očitavanje Meta-TEDS sadržaja upotrebom STWS servisa.

Na slici 2.6 je ilustrovano uvođenje prostih i nizovnih IDL tipova modula IEEE1451Dot0::Args, u WSDL element types opisan tabelom 2.4, a uz upotrebu XML Schema jezika. Primerom sa slike, definišu se prosti tipovi Uint16, Int32 i

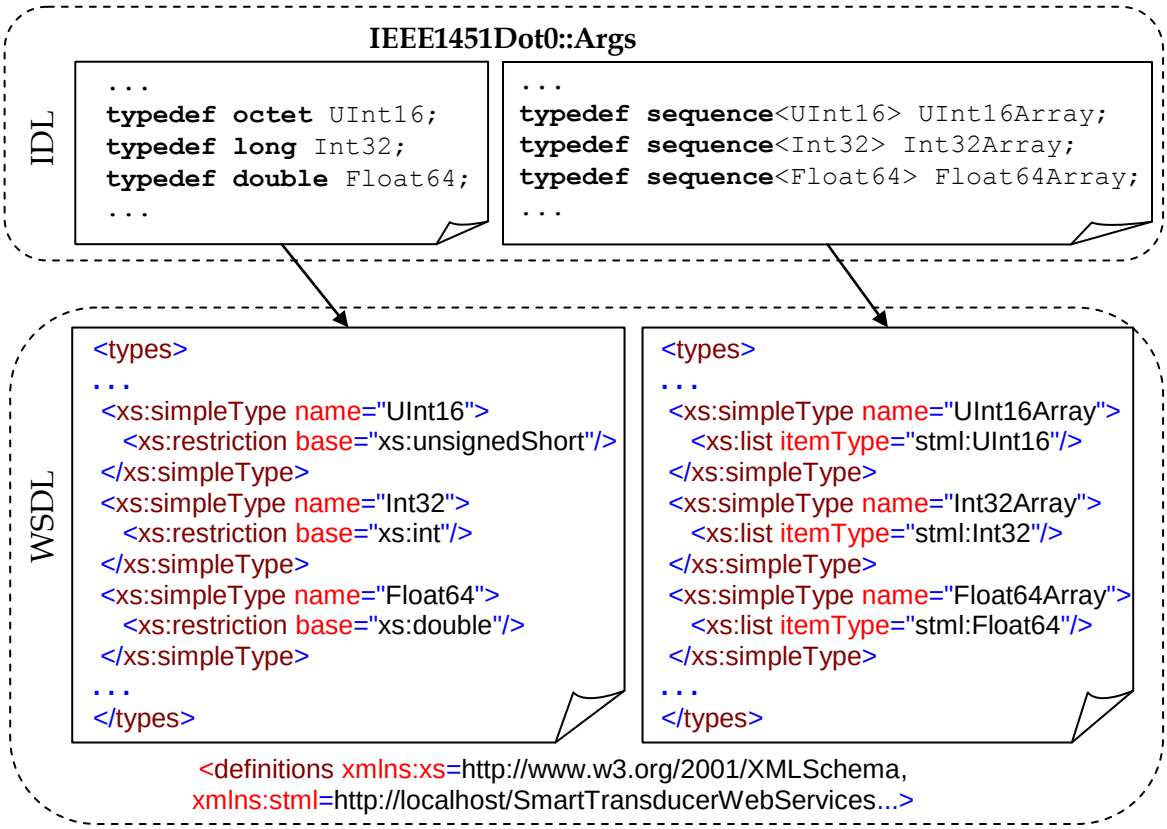
Float64, kao i njima pridruženi nizovni tipovi, u okviru odgovarajućeg XML imenskog prostora.

Tabela 2.4. XML struktura WSDL servisne definicije.

NAZIV XML ELEMENTA	OPIS
definitions	Definicija imena Web servisa i imenskih prostora je određena pomoću *definitions*, kao elementom najvišeg nivoa koji sadrži sve ostale elemente.
types	Definicija tipova podataka koji su relevantni za razmenu poruka između servera i klijenta. Uopšteno, ovde se definišu prosti i drugi tipovi definisani standardom, kao i složeni tipovi koji se koriste u klijent/server komunikaciji. Specijalno, ovim elementom se u XML uvode tipovi definisani IEEE 1451.0 *Args* modulom (slika 2.1).
message	Ime i format poruke koja se prosleđuje kao zahtev ili odgovor servisa se definiše ovim elementom. Pritom se koriste tipovi definisani elementom *type*.
portType	Deo WSDL-a u kome se navodi skup apstraktnih operacija Web servisa sa ulazno/izlaznim porukama definisanim pomoću *message* elementa. Specijalno, pomoću ovog elementa je moguće definisati i apstraktne operacije STWS servisa, bazirane na IEEE 1451.0 servisima pametnog pretvarača.
binding	Vezivanje apstraktnog porta za konkretnu realizaciju servisnih operacija, vrši se u okviru elementa *binding*. Na taj način se može navesti neki od ponuđenih *binding* stilova, kao i transportni protokol (npr. HTTP, SMTP) za prenos XML poruke. Stil koji se koristi dalje u radu podrazumeva **document/literal**, što znači da su poruke formatirane shodno XML Schema tipovima elementa *type*.
service	U okviru WSDL fajla se nalazi i element *service* koji služi za specificiranje adrese Web servisa, najčešće u formi URL-a.

Pomoću atributa WSDL elementa <*definitions*> uvodi se imenski prostor XML Schema jezika *xs*. Unutar *xs* imenskog prostora definisani su osnovni i nizovni IEEE 1451.0 tipovi posredstvom postojećih XML Schema tipova. Vrednosti atributa *name* predstavljaju imena 1451.0 tipova, dok su vrednosti atributa *base* tipovi Schema jezika (npr. *Uint16* i *xs:unsignedShort*). Rezultujući osnovni tipovi XML-a su deo *stml* imenskog prostora i referencirani su prilikom definisanja novih nizovnih tipova. Na taj način je stvorena osnova za prenos parametara putem STWS interfejsa, neophodnih za formiranje argumenata poziva IEEE 1451.0 metoda. Takođe, na osnovu datih tipova, formiraju se kompleksni tipovi za prenos specijalnih parametara (npr. vremenske reference) i generički niz *ArgumentArray*. Konačno, potrebno je definisati i kompleksan tip koji služi kao šablon za instanciranje XML poruka zahteva/odgovora STWS servera. Poruka XML zahteva koju prima STWS server mora da sadrži sve

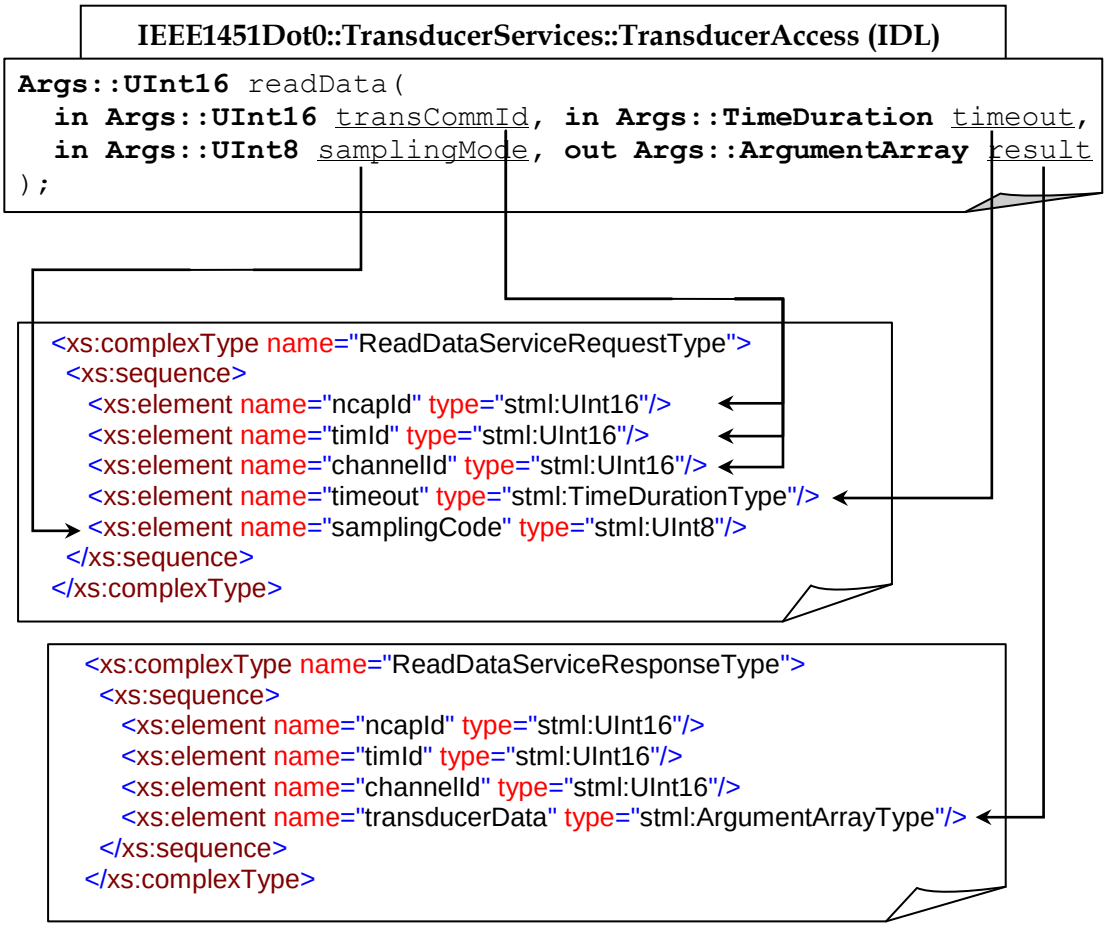
parametre koji su neophodni za pozivanje odgovarajuće metode IEEE 1451.0 sloja. Poruka XML odgovora mora da sadrži polja koja omogućavaju vraćanje rezultata odgovarajuće IEEE 1451.0 metode.



Slika 2.6. Uvođenje prostih i nizovnih tipova modula IEEE1451Dot0::Args u WSDL definiciju servisa upotrebom XML Schema jezika.

Na slici 2.7 su prikazani formati XML zahteva i odgovora koji se koriste za mrežni poziv IEEE 1451.0 metode *readData*. U tom slučaju, koristi se *ReadDataService* STWS sloja, koji obezbeđuje argumente poziva IEEE 1451.0 metode i prihvata povratne parametre.

Zahtev servisa koji je tipa *ReadDataServiceRequestType* prikazanog na slici, sadrži identifikaciju kanala čiji se podaci zahtevaju, a u okviru parametara *ncapId*, *timId*, *channelId*. Na osnovu ta tri parametra se određuje *transCommId*, odnosno broj komunikacione sesije pretvaračkog kanala, kao argument metode *readData*. Vrednosti *timeout*, *samplingMode* i *ArgumentArray* tipa se prenose direktno između STWS i 1451.0 sloja, kao na slici.



Slika 2.7. Definicija XML formata za prenos poruka zahteva/odgovora STWS servisa *ReadDataService*, a na osnovu IDL definicije servisne metode *readData* u okrivu IEEE 1451.0 sloja.

Zatim, putem *message* i *portType* elemenata definišu se apstraktne operacije servisa koje razmenjuju podatke u formatu određenom *type* elementom. Po pristizanju poruke zahteva servisa, posebna komponenta servera analizira sadržaj i poziva odgovarajući servis kome se prosleđuju parametri.

Kako bi definisani kompleksni tipovi i formalno postali deo poruke zahteva/odgovora određenog STWS servisa, potrebno je definisati element *message* kao na slici 2.8 (a). Pritom se kao deo poruke navodi već definisani kompleksan tip zahteva/odgovora prikazan slikom 2.7.

Element *portType* omogućava definiciju apstraktnog porta servisa u okviru WSDL jezika, koji sadrži sve potrebne servisne operacije. Na slici 2.8 (b) je dat primer definicije apstraktne operacije *ReadDataService*, koja podrazumeva razmenu ranije definisanih poruka, dok se ostale operacije definišu na sličan način.

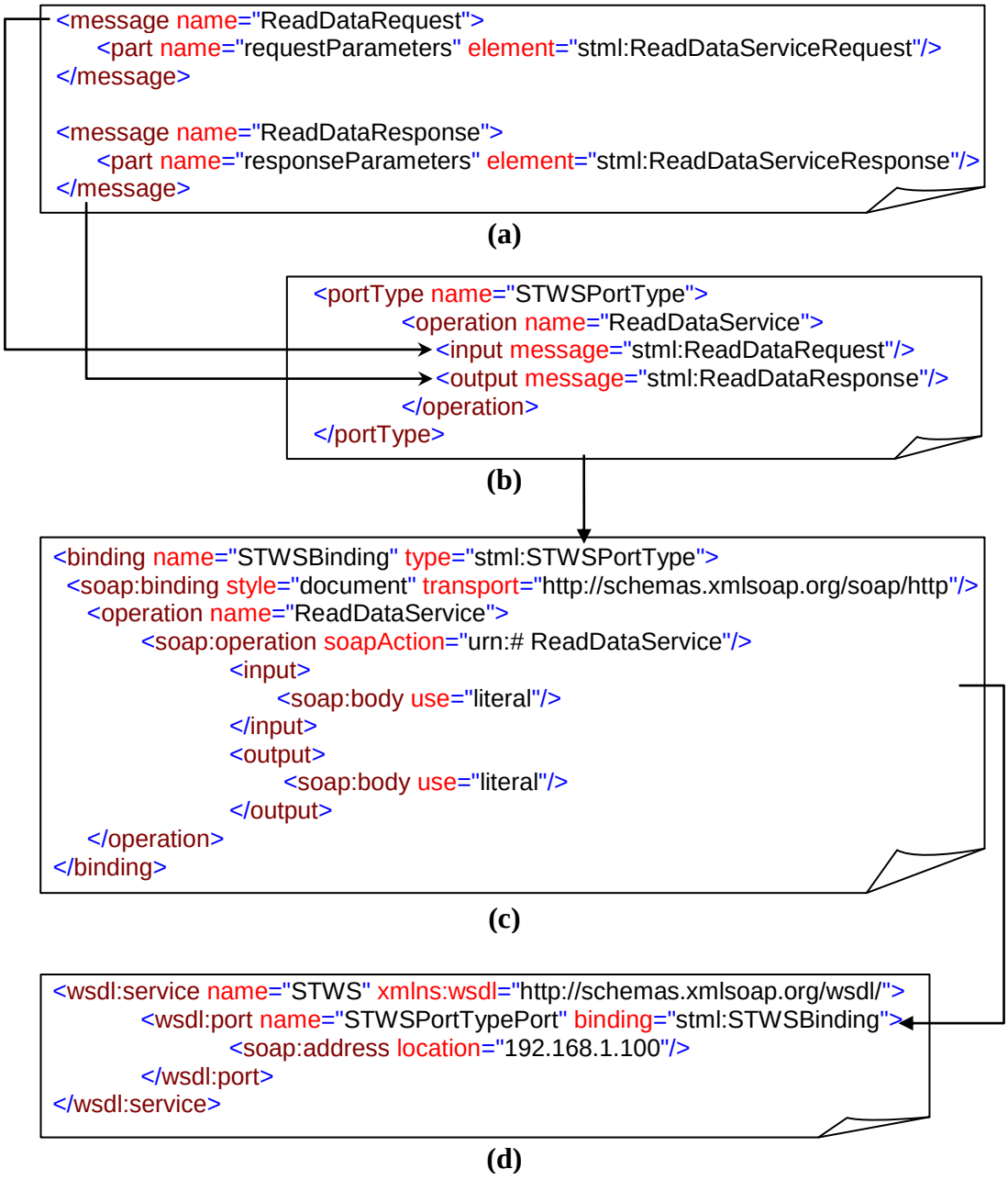
Kako bi se definisala razmena poruka upotrebom konkretnih protokola, koristi se *binding* element ilustrovan WSDL isečkom na slici 2.8 (c). Prema slici, kao protokoli za prenos unapred definisanih poruka koriste se HTTP i *Simple Object Access Protocol* (SOAP), kao i *document-literal* stil kodovanja. Time se obezbeđuje da telo SOAP zahteva/odgovora zapravo čini poruka u formatu koji je specificiran XML Schema definicijom.

Konačno, na slici 2.8 (d) se ilustruje vezivanje instance konkretnog porta za IP adresu. Fiksna IP adresa je prvenstveno namenjena upotrebi Web servisa u slučaju implementacije poslovne logike preduzeća. U slučaju STWS servisa postoji potreba za povremenim izmenama IP adrese, zbog dodavanja i uklanjanja pametnih pretvarača, što se reguliše u vreme izvršavanja programa.

Opisana definicija je u skladu sa WSDL 1.1 specifikacijom koja nailazi na bolju podršku dostupnih razvojnih okruženja. Prema WSDL 2.0 verziji specifikacija se pravi na sličan način, pri čemu su uvedene neke manje promene. Tako na primer umesto <*portType*> elementa postoji <*interface*> element, dok je <*message*> element potpuno izbačen. Umesto <*definitions*> elementa, koristi se <*description*>.

Sama definicija servisa ne bi imala tako značajan uticaj na implementaciju, da ne postoje gotovi generatori koji na osnovu WSDL specifikacije daju kao rezultat izvorni kod klijentske i serverske strane. Kao što je prikazano na slici 2.9, a u zavisnosti od tipa korišćenog generatora, moguće je dobiti izvorni kod u više programskih jezika na osnovu iste WSDL specifikacije. Prikazani slučaj podrazumeva projektovanje sistema odozgo na dole i podrazumeva da se kreće od WSDL specifikacije i završava izvršnom verzijom

programa. Takođe, moguće je uz pomoć odgovarajućih generatora, na osnovu izvornog koda, generisati WSDL specifikaciju.



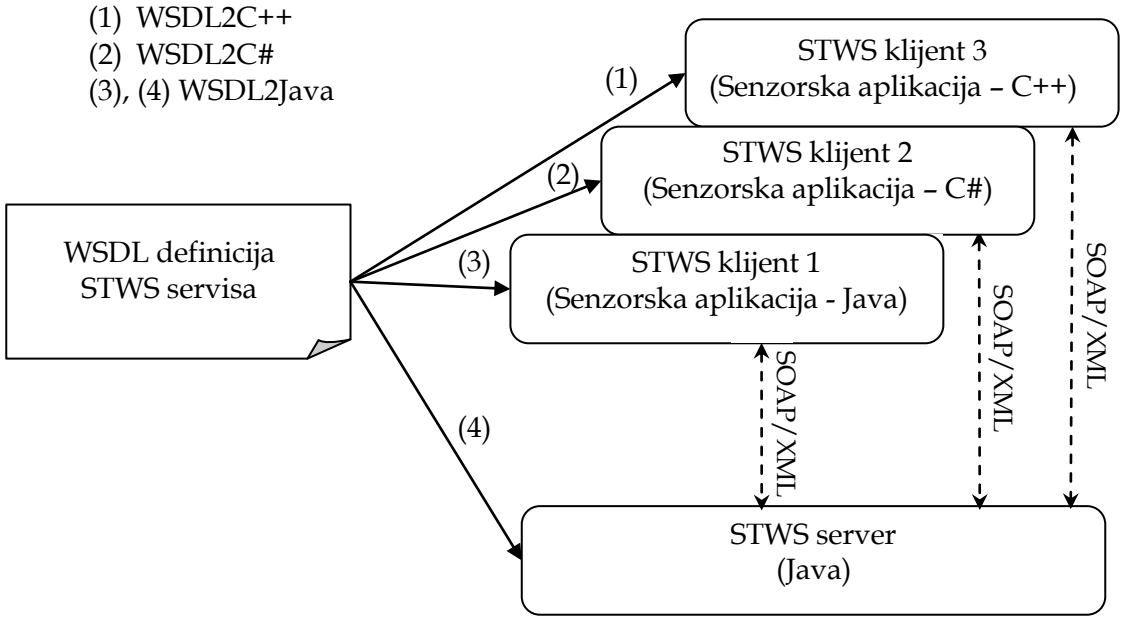
Slika 2.8. Definicija apstraktnih STWS servisa i dodeljivanje konkretnih protokola: (a) Definicija poruke (*message*) na osnovu ranije definisanih XML tipova; (b) Definicija tipa porta (*portType*) sa skupom servisnih operacija; (c) Vezivanje (*binding*) tipa porta za konkretne protokole i stil kodovanja podataka; (d) Dodeljivanje mrežne adrese instanci porta.

Slično kao u slučaju korišćenja IDL definicije, ako se uzme primer Java ili C++ programskog jezika, dobijaju se klase pomoću kojih se na jednostavan način vrše pozivi servisa na klijentskoj strani i vrši upis/čitanje sadržaja zahteva/odgovora servisa. Pri tome, postupak konverzije klasa zahteva i odgovora u XML poruke je automatizovan. Na sličan način u programskom jeziku C se u većini slučajeva vrši konverzija između strukture podataka i tekstualnog XML sadržaja. HTTP protokol predstavlja pogodnu opciju za transport tekstualnih XML poruka čiji je format definisan WSDL specifikacijom. Korišćenje servisnog steka podrazumeva da se uvodi dodatni sloj koji automatski tekstualni XML sadržaj primljen putem HTTP-a prebacuje u memoriju programa (deserijalizacija) i odgovarajući memorijski sadržaj prevodi u XML (serijalizacija) za slanje putem odgovarajuće HTTP metode. U tom slučaju programer ne mora da vodi računa o postupku setovanja HTTP zaglavlja i o formatiranju HTTP sadržaja koji se prenosi. Kao rezultat, server razmenjuje SOAP/XML poruke sa uredjajima koji su prijavljeni u mreži, pri čemu se HTTP koristi kao njihov transportni protokol, čime se dobija veći stepen automatizacije.

Postoji više načina na koji se može realizovati konkretan sistem prema prikazanom konceptu, pri čemu se sloboda dizajna ogleda u načinu integracije aplikacije za obradu podataka, dodavanju novih servisa i funkcionalnosti, kao i uvođenjem podrške za sinhronizaciju i konfiguraciju mreže.

Detaljnija analiza realizacije sistema je predmet ove teze i biće prikazana u vidu modela implementacije mreža pametnih pretvarača. Pri tome treba napomenuti da STWS servisi ne predstavljaju zamenu za OGC-SWE servise, već opisuju komplementarni sloj kojim se fizički uređaji efikasno povezuju sa OGC-SWE servisima. Povezivanje fizičkih uređaja sa OGC-SWE servisima je predstavljeno u radu [19], pri čemu se preko Interneta pristupa geografski dislociranim, posredničkim STWS servisima. STWS server je implementiran na platformi opšte namene, što je uobičajeno za ovakav tip servisa, dok će u ovoj

tezi biti predstavljen direktan pristup pametnom pretvaraču realizovanom na namenskoj računarskoj platformi sa integrisanim STWS interfejsom.



Slika 2.9. Ilustracija generisanja izvornog koda servisa za tri različita programska jezika na osnovu jedinstvene WSDL specifikacije.

2.4. Analiza postojećih rešenja i cilj istraživanja

Postojeća rešenja se baziraju na opštoj arhitekturi koja je prikazana slikom 2.4, odnosno na klasičnom klijent-server modelu. Dati model je baziran na IEEE 1451.0 standardu i podrazumeva realizaciju standardnih funkcionalnosti, upotrebu standardnih protokola i elektronskih specifikacija pametnog pretvarača. Servisno orijentisana arhitektura se javlja u vidu nadogradnje standarda upotrebom Web servisa, tako da se dolazi do većeg stepena interoperabilnosti. Upotreba STWS servisa omogućava dopremanje podataka iz fizičkih uređaja do nivoa mreže, kako bi bili procesirane od strane OGC-SWE ili drugih aplikacija višeg nivoa. Međutim, ponuđeni modeli se ne bave standardnim načinom konfiguracije mreže, automatizovanom razmenom podataka i sinhronizacijom pojedinih aktivnosti tokom izvršavanja distribuirane aplikacije.

Data prototipska rešenja se tipično implementiraju pema modelu sa slike 2.4 (a) i (b), na bazi mrežnog procesora koji se implementira na platformi opšte namene. Slučaj sa slike 2.4 (c) koji se bazira na namenskoj računarskoj platformi predstavlja poseban implementacioni izazov zbog toga što je potrebno integrisati sve komponente u jedinstveni NCAP-TIM uređaj. Zbog toga će u ovoj tezi kao osnova za razvoj pametnog pretvarača biti korišćen generički namenski modul koji odgovara slučaju sa slike 2.4 (c). Takođe, ideja je da se poslovi menadžmenta mreže izdvoje iz same aplikacije za obradu podataka u poseban mrežni čvor, tako da se dobije centralizovana komponenta za upravljanje mrežom.

Dakle, glavni cilj istraživanja se ogleda u unapređenu postojećih mrežnih modela, što se ostvaruje na bazi servisno orijentisane arhitekture i uz zadržavanje postojećih funkcionalnosti mreže. Kako bi centralni menadžerski čvor imao uniforman pristup pametnim pretvaračima i aplikacijama za obradu podataka, potrebno je uvesti izmene u modelu implementacije aplikacionih čvorova. Aplikacioni čvor se realizuje u formi virtuelnog pretvaračkog modula, tako da se funkcionalnostima aplikacije za obradu podataka pristupa preko interfejsa identičnog kao u slučaju pametnog pretvarača. Time se dobija uniforman pristup menadžerskog čvora ostalim uređajima, gde se kao osnova plug & play funkcionalnosti koriste elektronske specifikacije. Forma virtuelnog pretvaračkog modula je osmišljena tako da omogući integraciju različitih mrežnih entiteta poput displeja, tastature, alogritama i drugih entiteta. Navedene izmene su osnova za uvođenje novog modela implementacije servisno orijentisanih mreža pametnih pretvarača koji će biti opisan u ovoj tezi. Poseban kvalitet novog modela se ogleda i u upotrebi aktivnih komponenti za automatizovanu razmenu podataka između mrežnih čvorova. Takođe, upotreba vremenske reference unutar modela podataka i uvođenje servisa za vremensku sinhronizaciju predstavljaju osnovu za uvođenje karakteristika sistema za rad u realnom vremenu.

Značaj novog modela se ogleda u pružanju podrške za implementaciju distribuiranih aplikacija upotrebom komercijalnih računarskih mreža. Novi model obezbeđuje efikasnu integraciju heterogenih uređaja i omogućava brz razvoj novih aplikacionih entiteta. Prilikom razvoja novog aplikacionog entiteta, dizajner je u velikoj meri rasterećen po pitanju poslova menadžmenta mreže i komunikacije, tako da je stvorena osnova za brz razvoj prototipa. Detalji novog modela i njegova verifikacija putem studija slučaja će biti prikazani u nastavku teze.

3. SERVISNO ORIJENTISANI MODEL RAZMENE PODATAKA I KONFIGURACIJE U MREŽAMA PAMETNIH PRETVARAČA

Prethodno opisana arhitektura koja se zasniva na klijent-server komunikaciji ne pruža standardni model konfiguracije i automatizovane razmene podataka u mreži pametnih pretvarača. Naime, upravljanje mrežom obezbeđuje klijentski čvor na kome se istovremeno izvršava i aplikacija za obradu mernih/upravljačkih podataka. Razdvajanjem poslova menadžmenta i aplikacionih poslova u zasebne mrežne čvorove, dobija se novi model implementacije koji omogućava nezavisan razvoj ove dve komponente. U tom slučaju, upravljanje se odvija pomoću centralnog menadžerskog čvora koji koordinira radom svih ostalih mrežnih entiteta. Takođe, novim modelom se aplikacija za obradu podataka prevodi iz tradicionalne klijentske u serversku formu, tako da postaje pasivna komponenta poput samog pametnog pretvarača, odnosno entitet kojim takođe treba upravljati. Na taj način se obezbeđuje da se čitava mreža sa stanovišta centralnog menadžerskog čvora vidi kao skup serverskih uređaja, pri čemu su servisi pametnog pretvarača definisani kao i ranije (STWS servisi), ali ostaje problem nedefinisanih servisa za pristup aplikacijama za obradu podataka. Pri tome se nameće rešenje uvođenja uniformnog pristupa pametnim pretvaračima i aplikacijama za obradu podataka, što znači da se i za aplikacione servise usvajaju STWS servisi. Takvim razvojem modela se dolazi do koncepta virtuelnog pretvaračkog modula koji poseduje isti mrežni interfejs kao i standardni pretvarački modul, ali se umesto fizičkih kanala (senzora i aktuatora) implementiraju algoritmi za obradu podataka i ulazno/izlazni uređaji poput displeja za prikaz rezultata obrade. U nastavku je dat opis realizacije generičkog modula pametnog pretvarača, virtuelnog pretvaračkog modula i centralnog menadžerskog čvora, na osnovu definicije uniformnog STWS interfejsa. Takođe, dat je primer implementacije mreže na osnovu novog modela i sa upotrebom aktivne komponente na strani

menadžerskog čvora za automatizovanu razmenu podataka između pametnih pretvarača i virtuelnih pretvaračkih modula.

3.1. Prototip generičkog modula pametnog pretvarača

Model pametnog pretvarača opisan IEEE 1451.0 standardom omogućava upotrebu različitih platformi za implementaciju od 8-bitnih mikrokontrolera, pa sve do personalnih računara koji imaju funkciju NCAP-a. U dostupnim radovima se često nailazi na prototipska rešenja u kojima se mrežni procesor NCAP opremljen STWS servisima implementira na platformi opšte namene ili se razmatra implementacija Web servisa na namenskim računarskim platformama nevezano za IEEE 1451.0 standard. Međutim, savremeni mikrokontroleri sa mogućnošću mrežne komunikacije se mogu koristiti kao osnova za realizaciju uređaja koji kombinuju funkcionalnosti NCAP i TIM modula prema modelu sa slike 2.4 (c).

Početnu osnovu prototipskih mreža koje će biti prikazane u ovoj tezi čine STWS servisi implementirani na mikrokontroleru koji je baziran na ARM arhitekturi. Korišćen je 32-bitni mikrokontroler LPC1768, kompanije NXP [20], sa Cortex-M3 procesorskom jedinicom radne frekvnecije do 100MHz. Cortex-M3 procesor podržava protočnu obradu sa 3 nivoa izvršavanja operacija i Harvard arhitekturu sa odvojenim lokalnim linijama instrukcija i podataka, kao i posebnom magistralom za periferije. Osim toga, mikrokontroler poseduje i 512kB flash memorije, 64kB memorije podataka, kao i Ethernet MAC, 12-bitne ADC, 10-bitni DAC i druge periferije.

Za korišćenje periferija datog mikrokontrolera, dostupne su i gotove biblioteke [21]. Tako je moguće realizovati HTTP server korišćenjem odgovarajućeg API-ja, pa je ova opcija iskorišćena za realizaciju transportnog protokola za SOAP/XML poruke. Ipak, u trenutku izrade ove teze nije postajala mogućnost efikasne izrade celokupnog servisnog steka samo na osnovu postojeće biblioteke, a na osnovu WSDL definicije. Kako bi se upotpunio servisni stek, korišćen je gSOAP [22], kao generator izvornog koda servisa. Alat gSOAP omogućava prevođenje WSDL specifikacije u C/C++ izvorni kod

korišćenjem kompajlerskih tehnologija, priznat je u industriji i kompatibilan sa preporukama za interoperabilnost između uređaja različitih proizvođača. Preporuke se u ovom slučaju daju od strane organizacije *Web Services Interoperability Organization* (WS-I), u vidu profilnih specifikacija i alata za praktičnu realizaciju Web servisa. Međutim, rezultujući izvorni kod je prvenstveno namenjen određenoj grupi operativnih sistema, zbog čega je neophodno portovanje takvog koda na ARM mikrokontroler, na kome se u ovom slučaju koristi servisni stek bez operativnog sistema.

Prvobitno je definisan WSDL fajl na bazi IEEE 1451.0 operacija slično kao u radu [19], pri čemu je dati proces već detaljnije ilustrovan u okviru potpoglavlja 2.3. Korišćeni su gSOAP alati za dobijanje izvornog koda STWS servisa na osnovu WSDL specifikacije za Linux platformu. Prilikom kreiranja izvršne verzije programa u komandnoj liniji Linux-a, upotrebom **gcc** paketa i gotovog izvornog koda, automatski se linkuju i standardne biblioteke soketa. Za potrebe portovanja dobijenog izvornog koda na ARM mikrokontroler, korišćeno je razvojno okruženje Keil uVision, pri čemu je izvorni kod servisa modifikovan i preveden u izvršnu verziju.

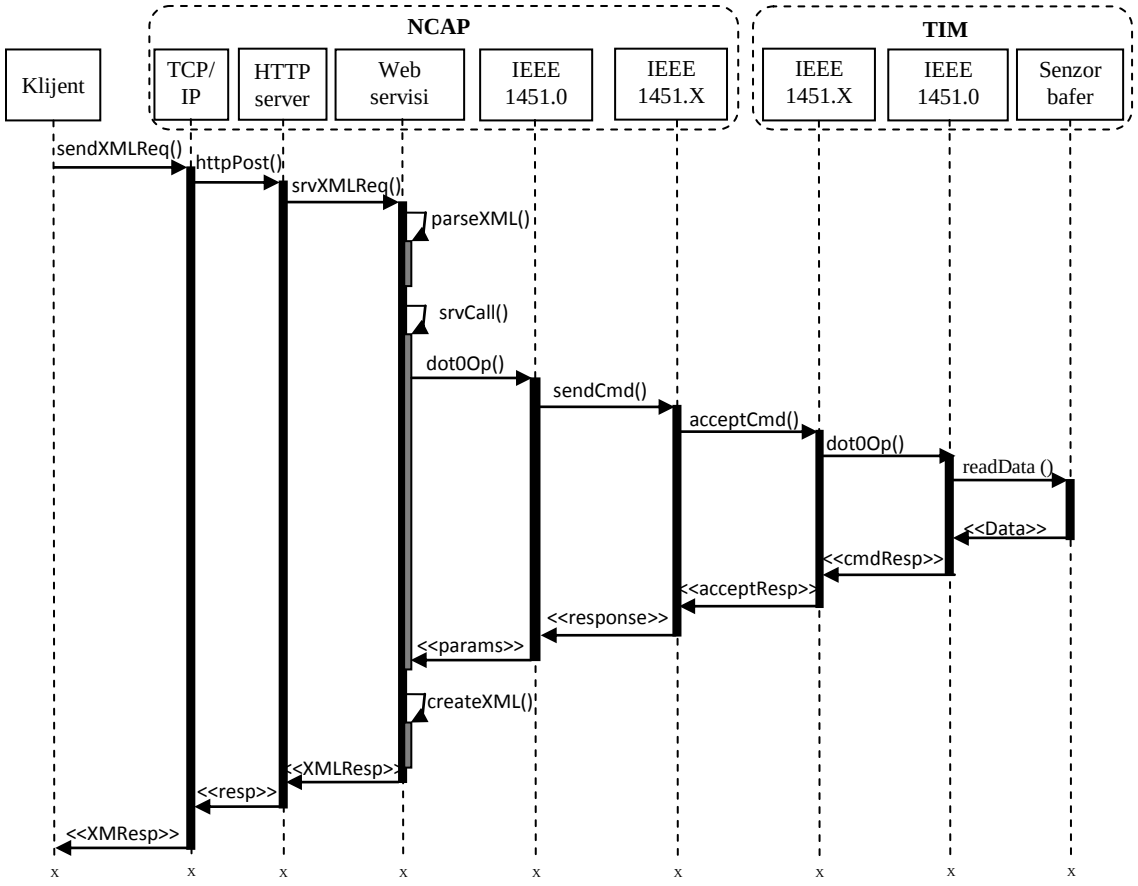
Kada se isti kod kompajlira i povezuje u okviru Keil uVision okruženja, dobijaju se greške usled neslaganja definicija tipova podataka i usled nedostatka sistemskih fajlova sa definicijama koje se odnose na rad sa soketima. U takvoj situaciji, nameće se rešenje korišćenja bafera umesto soketa, pa je u tom pravcu i izvršena promena izvornog koda servisa. Pošto je ovakvu promenu potrebno izvršiti pri svakoj promeni WSDL fajla i upotrebi gSOAP-a, proces portovanja je automatizovan korišćenjem BASH i PERL skripti. Nakon portovanja koda dobijenog pomoću gSOAP-a, STWS servisni stek je uspešno kompletiran upotrebom postojećeg HTTP servera za transport SOAP/XML poruka korišćenjem HTTP POST metoda. Time je zapravo i ostvaren cilj integracije STWS servera u pretvaračkи modul prema modelu sa slike 2.4 (c).

Jedna od prepreka prilikom usvajanja IEEE 1451 familije standarda jeste i njegova kompleksnost. Pri tome, već je rečeno da izabrani mikrokontroler,

pored Ethernet interfejsa za mrežnu komunikaciju, poseduje i različite periferije za implementaciju mernog/upravljačkog interfejsa (ADC i DAC periferije). Zbog toga je u okviru sledećeg koraka integrisana i funkcionalnost IEEE 1451.0 sloja i TEDS, tako da se dobije kombinovani NCAP-TIM modul. Na taj način se dobija generički modul kao osnova za implementaciju različitih pametnih pretvarača. Povezivanjem odabranih pretvarača i pratećih kola na ulaz ADC i DAC, kao i odgovarajućim popunjavanjem TEDS struktura dobija se konkretan pametni pretvarač.

Cilj ovakve kompaktne realizacije jeste eliminacija IEEE 1451.X sloja i pratećih SW/HW komponenti. Na primer, nije potrebno implementirati *Codec* interfejs za kodovanje/dekodovanje komandi, kontrolu prenosa paketa i timeout logiku na NCAP strani za slučaj kada se čeka odgovor TIM-a. Mana ovakvog pristupa je svakako smanjena modularnost samog pametnog pretvarača, pri čemu se gubi plug & play mogućnost na nivou IEEE 1451.X interfejsa. Na slikama 3.1 i 3.2 je ilustrovana ušteda koja se dobija prilikom očitavanja merne vrednosti senzora iz bafera senzora TIM modula. Na slici 3.1 je prikazana komunikaciona sekvenca za slučaj pametnog pretvarača sa IEEE 1451.X komunikacionim stekom, dok je u slučaju 3.2 prikazan kombinovani NCAP-TIM modul sa jednostavnijom sekvencom koji će biti korišćen u svim prikazanim modelima. Prvobitno, klijent šalje XML zahtev putem TCP/IP i HTTP protokola, koji se zatim parsira na sloju Web servisa i poziva se odgovarajući servis. Servis zatim poziva operaciju *dot0* sloja (IEEE 1451.0 sloj), nakon čega se komanda za čitanje mernih podataka prosleđuje IEEE 1451.X sloju, kao što je prikazano u slučaju sa slike 3.1. Poruka se zatim šalje TIM modulu, izvršava se *dot0* servis za čitanje podataka, nakon čega se izvršava sekvenca za slanje podataka klijentu. Pritom, podrazumevano je da se čitanje senzora i upis u aktuator vrši kontinualno bez slanja *trigger* signala konverzije na strani senzora, sa vremenskom periodom koja zavisi od implementacije konkretnog pretvaračkog kanala. Startovanje svake konverzije na strani senzora se inicira softverski nakon isteka zadatog perioda, pri čemu se nova merna

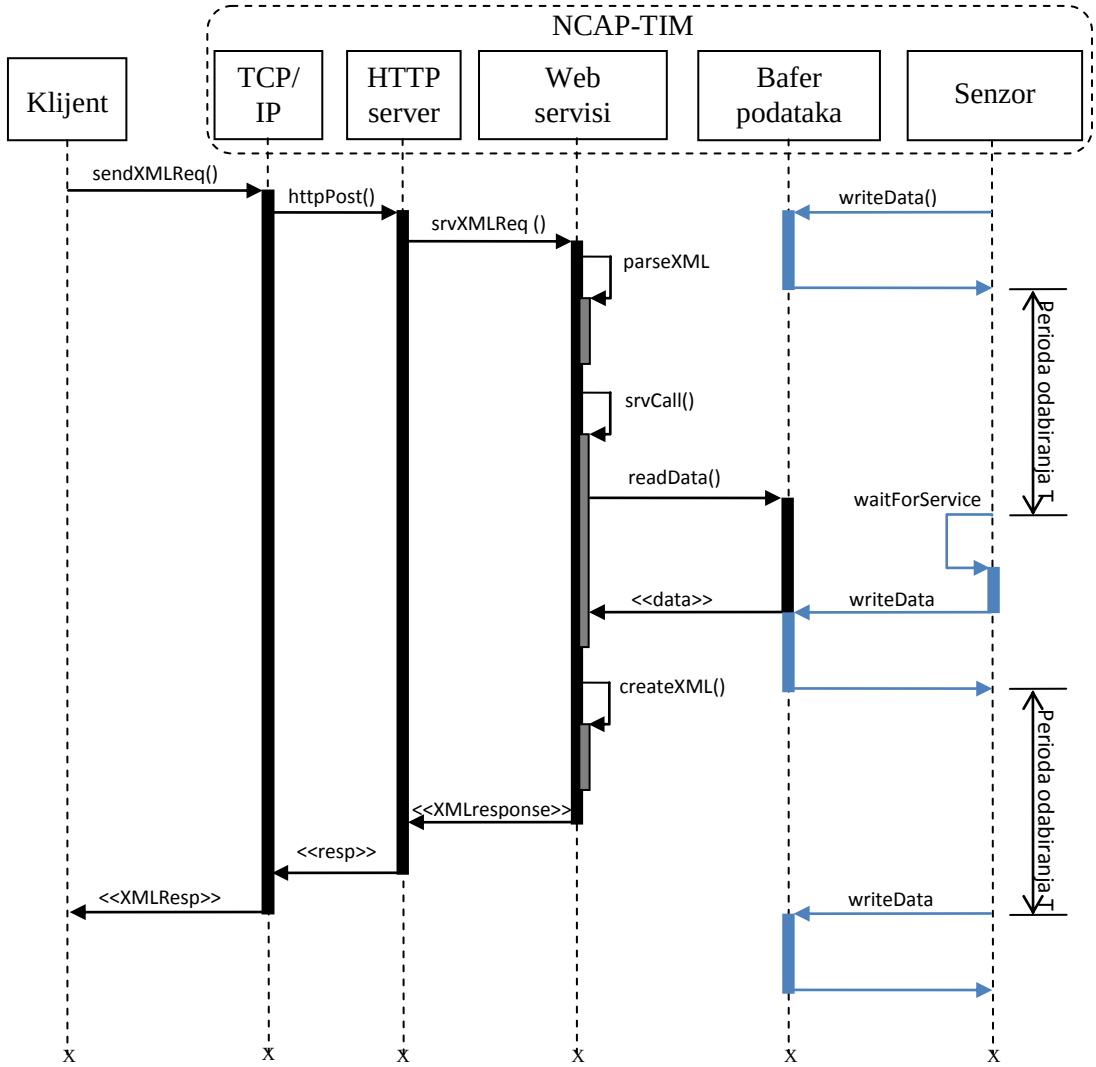
vrednost smešta u FIFO bafer ili se iz bafera aplicira kontrolni odbirak u slučaju aktuatora. U slučaju sa slike 3.2 ilustrovana je sekvenca poziva operacija za slučaj implementacije pametnog pretvarača u vidu jednog modula. Pritom, apstrakovane su funkcije IEEE 1451.0 sloja, ali je prikazan proces periodičnog upisa mernih podataka u bafer čiji sadržaj se očitava tokom izvršavanja servisa. Prikazani Web servisi su zapravo STWS servisi dobijeni pomoću gSOAP generatora izvornog koda, a na osnovu postojećeg WSDL-a.



Slika 3.1 Dijagram sekvenci poziva operacija u slučaju očitavanja mernih podataka senzora, pri čemu su NCAP i TIM su zasebni moduli koji komuniciraju putem IEEE 1451.X interfejsa;

Konzistentnost pistupa baferu se na odabranom mikrokontroleru obezbeđuje zabranom prekida u kritičnim sekcijama programa. Proces periodičnog upisa podataka u bafer postoji i u slučaju sa slike 3.1, pa se može zaključiti da postoji ušteda u koracima u odnosu na tipičnu realizaciju gde su

NCAP i TIM razdvojeni. Takođe, u slučaju sa slike 3.1 postoji dodatni poziv uslužne funkcije IEEE 1451.X sloja za kodovanje/dekodovanje poruke zahteva, pre njenog slanja TIM modulu. Eliminisanjem IEEE 1451.X sloja se dobija jednostavniji model implementacije pametnog pretvarača, pri čemu se zadržava postojeći mrežni interfejs. Pored toga, STWS server je realizovan na namenskoj računarskoj platformi što omogućava direktan pristup funkcijama pretvarača preko mreže.



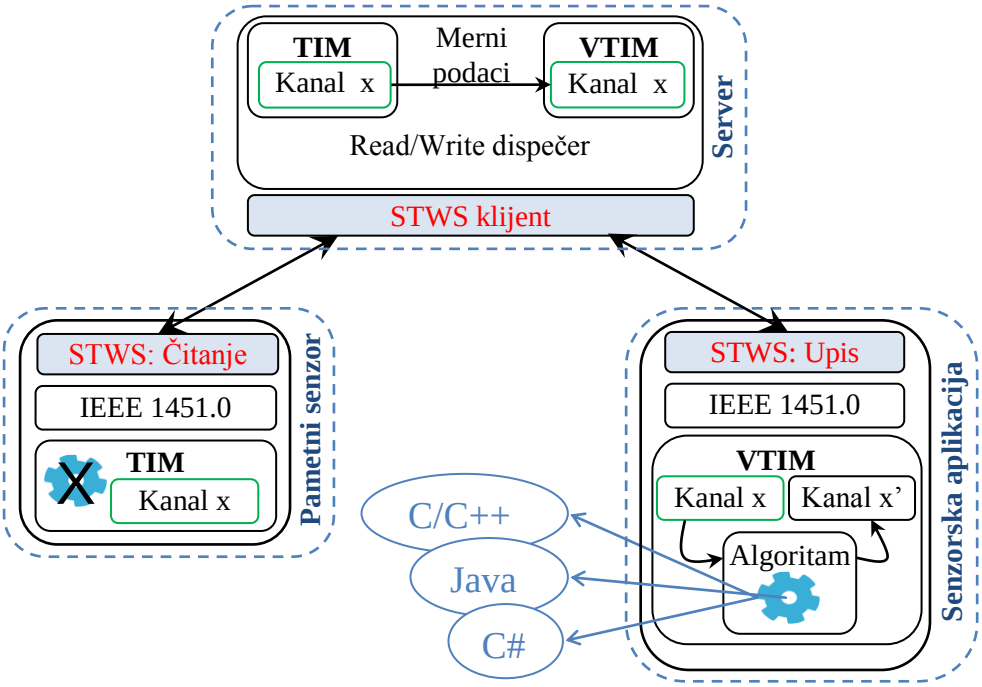
Slika 3.2 Dijagram sekvenci poziva operacija u slučaju očitavanja mernih podataka senzora, pri čemu se pametni pretvarač realizuje u vidu jednog modula koji kombinuje NCAP i TIM funkcionalnosti.

3.2. Koncept virtuelnog pretvaračkog modula

Uniforman način pristupa obezbeđen pomoću WSDL definicije uvodi mogućnost ostvarivanja interoperabilnosti u servisno orijentisanoj mreži pametnih pretvarača. Kao sledeći korak, moguće je uvesti novi entitet kome se pristupa preko već definisanih Web servisa. Kako bi se ostvarila simetrija u pristupu pametnim pretvaračima i njihovim aplikacijama uvodi se koncept virtuelnih pretvarača [23].

3.2.1. VTIM arhitektura

Virtuelni pametni pretvarač predstavlja pristup u modelu implementacije u kome se aplikacija za obradu podataka ili neki drugi entitet oprema STWS interfejsom, tako da podatke dobija putem STWS servisa (slika 3.3) [24]. Ovakva postavka zahteva uvođenje centralnog autoriteta koji vodi evidenciju o prijavljenim uređajima i koordinira rad mreže. Centralni autoritet se implementira u formi menadžerskog čvora koji na uniforman način, preko STWS interfejsa, pristupa svim ostalim čvorovima mreže. Menadžerski čvor putem grafičkog korisničkog interfejsa daje mogućnost prikaza TEDS sadržaja različitih uređaja operateru i obezbeđuje konfiguraciju, kao i aktivnu komponentu za automatizovanu razmenu podataka između povezanih kanala. Tokom regularnog rada sistema, aktivna komponenta periodično čita podatke izvorišnog pretvaračkog kanala i upisuje ih u odredišni kanal. Pritom, implementacija dispečer komponente se zasniva na iteraciji kroz listu poslova distribuiranih u vremenu. Pristup namenskom, kao i virtuelnom pametnom pretvaraču, od strane aktivne komponente (*scheduler*-a) centralnog čvora obezbeđuje se slojem STWS klijenta. Menadžerski čvor se u ovom slučaju ponaša kao server sa stanovišta operatera, tako da sadrži i klijentski i serverski STWS interfejs, dok virtuelni/konkretni pametni pretvarači poseduju samo STWS serverski interfejs. Primerom sa slike se periodično očitava merni kanal *x* TIM modula i dobijeni podaci se upisuju u kanal x virtuelnog pretvaračkog modula VTIM (Virtual TIM), odnosno u kanal virtuelnog aktuatora.



Slika 3.3 Arhitektura VTIM mrežnog čvora za integraciju algoritma obrade podataka u mrežu pametnih pretvarača.

Umesto apliciranja odbiraka kao u slučaju realnog aktuatora, odbirci se prosleđuju algoritmu, tako da se nakon obrade rezultujući odbirci upisuju u kanal virtuelnog senzora x'. Virtuelnim i fizičkim kanalima, kao i TIM i VTIM TEDS komponentama se pristupa pomoću istog seta IEEE 1451.0 operacija. Prilikom poziva STWS servisa, dispečerska komponenta servera razmenjuje platformski neutralne SOAP/XML poruke sa uređajima u mreži. Svaki od uređaja dobija IP adresu dinamički putem DHCP protokola, što nije slučaj kod klasične upotrebe Web servisa. Web servisi se tipično nalaze na fiksnoj IP adresi, u skladu sa biznis procesima preduzeća, dok se u ovom radu upotrebljavaju kao dinamičke komponente u mreži pametnih pretvarača.

Pored uniformnog pristupa koji olakšava realizaciju menadžerskog čvora, VTIM koncept omogućava dislokaciju funkcija sa pametnog pretvarača na VTIM čvor. Tako se dobijaju manje strogi softverski/hardverski zahtevi po pitanju realizacije pametnog pretvarača, pri čemu se TEDS parametri autokorekcije i konverzije prebacuju sa TIM-a na VTIM zajedno sa pratećim

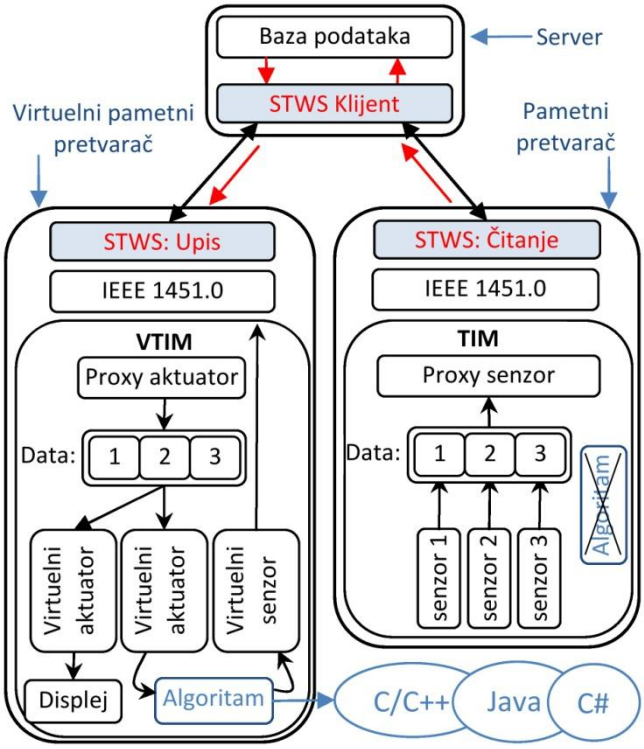
procesiranjem. U tom slučaju, pametni pretvarač sadrži minimalan set TEDS podataka, dovoljan da omogući otkrivanje kanala, njihov opis i čitanje/upis. Kada se na strani menadžera uspostavi putanja podataka i startuje komunikacija, TIM modul obezbeđuje „sirove“ (nekorigovane) podatke koji se obrađuju na dislociranom VTIM modulu. VTIM algoritam se tada implementira na platformi opšte namene u proizvoljnom programskom jeziku.

3.2.2. Primer konfiguracije VTIM izlaznog uređaja

Dislokacijom operacija pametnog pretvarača poput konverzije mernih podataka u fizičku veličinu i kompenzacione funkcije, mogu se pojednostaviti hardverski zahtevi za realizaciju pametnog pretvarača na namenskoj platformi ograničenih resursa i optimizovati njegova potrošnja. Međutim, mogućnosti realizacije nisu ograničene na razmatranje algoritama, uzimajući u obzir da je moguće integrisati i druge entitete u servisno-orijentisanu mrežu korišćenjem VTIM modela.

Primer realizacije displeja u formi VTIM mrežnog čvora je opisan u [23], pri čemu se konfiguracija displeja obavlja uvođenjem posebnih aplikacionih TEDS. Dati primer, ilustrovan slikom 3.4, prikazuje ideju integracije displeja gde se upis vrednosti vidi kao upis u virtuelni aktuator VTIM-a. Prema standardu IEEE 1451.0, korišćenje posredničkih (proxy) kanala olakšava očitavanje ili upis, sa obzirom da se adresiranjem jednog kanala dobijaju vrednosti svih njemu pridruženih kanala. Ova činjenica se može upotrebiti čak i za koegzistenciju displeja i algoritma u okviru jednog VTIM-a.

U opštem slučaju, upis u TEDS komponentu se vrši pozivanjem odgovarajućeg STWS servisa, što u slučaju VTIM-a sa slike 3.4 podrazumeva operacije poput konfiguracije displeja ili izbor odgovarajućeg algoritma. Sa obzirom na to da TEDS formati displeja i algoritma nisu definisani IEEE 1451.0 specifikacijom, potrebno je usaglasiti njihov sadržaj u vidu specijalnih aplikacionih TEDS komponenti.

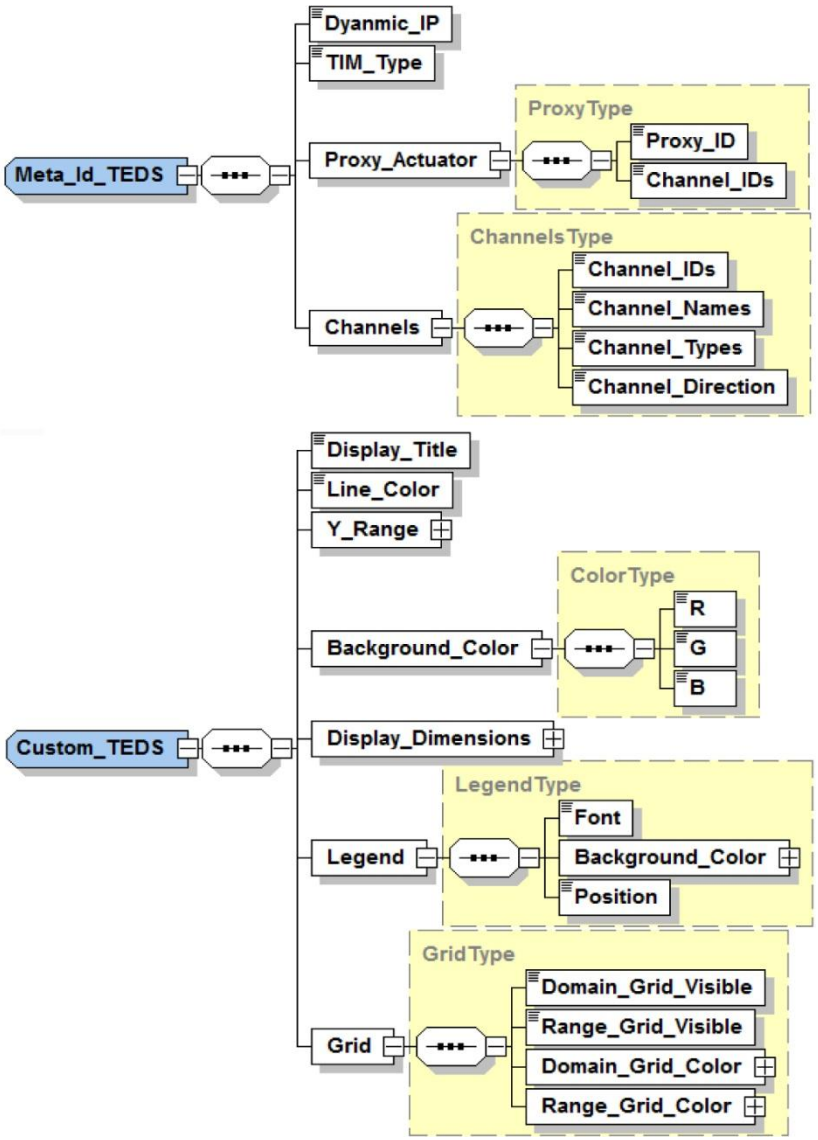


Slika 3.4 Integracija algoritma i displeja u formi VTIM-a u servisno orijentisanoj mreži pametnih pretvarača.

Na slici 3.5 je prikazan primer TIM TEDS formata (Meta_Id_TEDS) koji može da poseduje svaki TIM, kao i TEDS format pridružen isključivo virtuelnom aktuatoru na kome se nalazi displej (Custom_TEDS). Dijagrami predstavljaju XML Schema šablone koji se dodaju u WSDL definiciju za potrebe prenosa TEDS podataka putem Web servisa. Prvi dijagram sadrži podatke o tipu uredjaja (virtuelni ili konkretni modul), dodeljenu IP adresu, podatke o tipu kanala (aktuator/senzor), kao i opis proxy konfiguracije. TEDS format u donjem delu slike se odnosi na parametre prikaza displeja, poput naslova, dimenzija displeja, boje pozadine, prikazivanja grida i slično.

U nastavku je dat primer servisno-orijentisane konfiguracije jednog VTIM modula na kome se nalaze tri displeja (slika 3.6). Svaki displej prikazuje tri vrednosti: temperaturu vazduha, tačku rose (dew point temp.)i temperaturu podloge. Tačka rose se približno može izračunati na osnovu trenutne temperature i relativne vlažnosti vazduha i predstavlja vrednost temperature

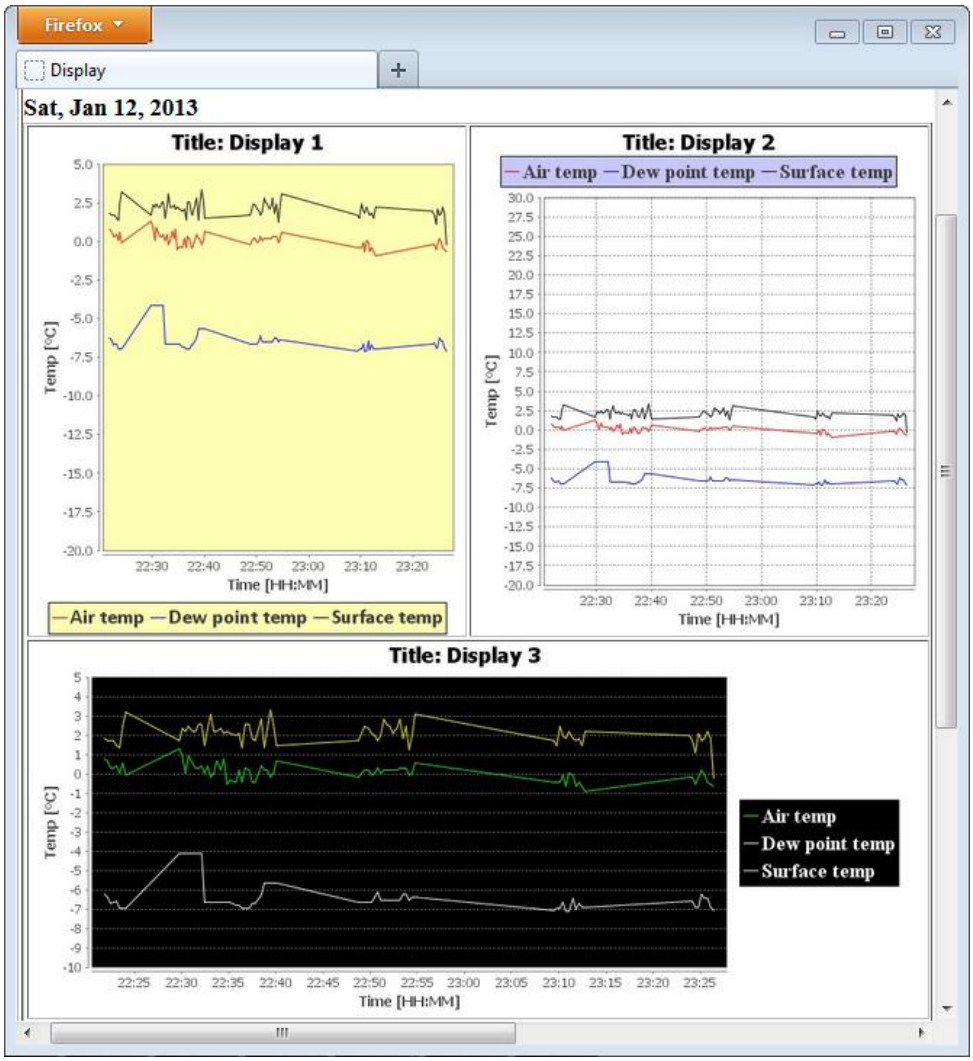
na kojoj vazduh postaje zasićen vodenom parom. Pametni senzor iz [25] je korišćen za dobijanje mernih vrednosti, dok je algoritam u vidu posebnog VTIM-a realizovan na platformi opšte namene i služi za računanje tačke rose. Krajnji rezultat se upisuje u bazu VTIM-a koji sadrži displej komponentu i koji je takodje realizovan na platformi opšte namene.



Slika 3.5 Primer identifikacione TIM TEDS komponente (gore) i TEDS-a pridruženog virtuelnom aktuatoru za ispis displeja (dole).

Server je implementiran u Java jeziku, pri čemu je korišćena Java DB (Derby) baza podtaka. Paralelno sa glavnom niti izvršavanja programa,

izvršava se i nit koja sadrži "timeout" logiku. U slučaju detekcije otkaza glavne niti, dispečer se može ponovo startovati i pokrenuti razmena podataka na osnovu parametara konekcije sačuvanih u bazi podataka. Pri tome, otkaz sistema utiče samo na poslove koji se trenutno nalaze u listi dispečera.



Slika 3.6 Prikaz VTIM displeja, realizovanog na PC platformi, za tri različite instance aplikacionih TEDS: (gore-levo) Displej 1 prikazuje podatke bez grid-a, sa opsegom *y*-ose [-20, 5]; (gore-desno) Displej 2 prikazuje podatke sa horizontalnim i vertikalnim grid-om, sa opsegom *y*-ose [-20, 30] i belom pozadinom; (dole) Displej 3 prikazuje podatke sa horizontalnim grid linijama, crnom pozadinom i opsegom y-ose [-10, 5].

Kritičan parametar koji odredjuje broj poslova u listi jeste broj povezanih uređaja i perioda očitavanja podataka, tako da se može smatrati da uređaj uvek

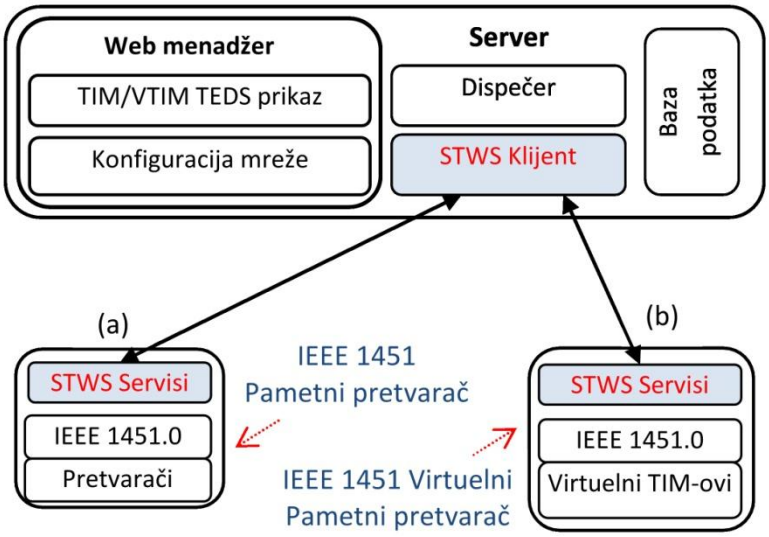
poseduje dovoljnu količinu memorije. Izvršavanje jednog posla je vremenski zahtevno sa obzirom na to da se vrše sinhroni pozivi Web servisa u Internet ili LAN mreži. Ubacivanje prevelikog broja poslova sa takvim kašnjenjem u listu može dovesti do nemogućnosti razmene podataka u zadatom vremenskom roku. Dati problem se može donekle ublažiti uvođenjem asinhronih poziva servisa. Pomenuti problem skalabilnosti može biti rešen raspoređivanjem uređaja na više međusobno povezanih menadžerskih servera. Time se može dobiti i dodatna podrška u slučaju otkaza jednog od njih, odnosno mogućnost prebacivanja poslova na druge aktivne servere.

Na slici 3.6, koja daje prikaz PC VTIM displeja, realizovanog u Javi, po upisu različitih konfiguracionih TEDS prema formatu sa slike 3.5, sva tri displeja prikazuju isti sadržaj iz postojeće baze podataka, pri čemu svaki od njih formatira prikaz na osnovu sopstvenog TEDS sadržaja. Svaki displej je podešen tako da prikazuje poslednjih 100 mernih vrednosti iz baze, pri čemu opseg x-ose za prikaz vremena zavisi od periode uzimanja uzorka i od trenutka startovanja merenja. Skup parametara sa slike 3.5 koji se mogu menjati je zadržan zbog jednostavnosti. Dati skup se može proširiti prema potrebi korisnika uvođenjem novih mogućnosti poput parametara za izbor vremenskog opsega za prikaz mernih vrednosti, izbor odgovarajućih slova za ispis naslova osa, uvođenje markera i izbor tipa dijagrama.

3.3. Menadžerski mrežni čvor

Model komunikacije u mreži sa centralnim menadžerom dat je slikom 3.7, pri čemu je u slučaju (b) prikazan virtuelni pametni pretvarač [24]. Svaka aplikacija (ili algoritam) za obradu podataka logički pripada zasebnom VTIM modulu. Na taj način je obezbedjeno da server praktično u mreži vidi samo pametne pretvarače, pri čemu se isti set servisa izvršava na virtuelnom pametnom pretvaraču i konkretnom pametnom pretvaraču prikazanom u slučaju (a). Osnovne komponente servera su sledeće: Web menadžer koji obezbeđuje GUI za prikaz TIM/VTIM TEDS sadržaja i omogućava konfiguraciju mreže; baza podataka za čuvanje TEDS sadržaja, podataka o

konfiguraciji mreže i ostalih podataka; kao i dispečer komponenta za procesiranje liste vremenski periodičnih i aperiodičnih poslova.

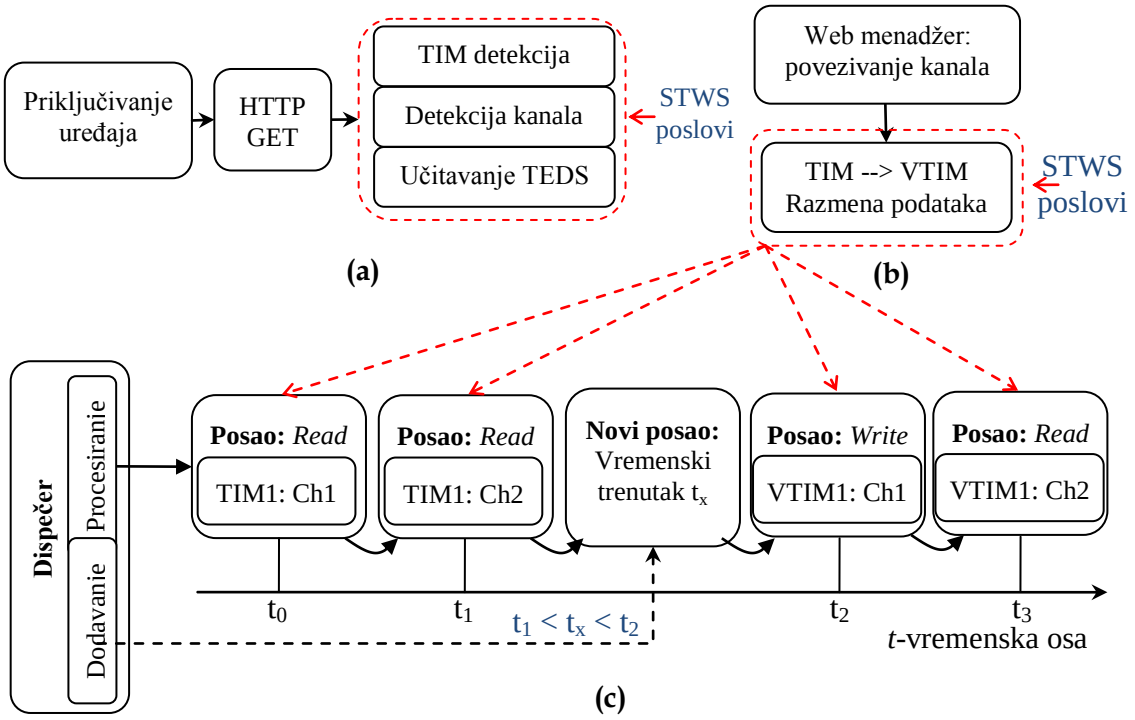


Slika 3.7 Model komunikacije za razmenu podataka pretvarača, posredstvom centralnog menadžerskog čvora.

Uredjaji se automatski prijavljuju po uključenju u mrežu slanjem svoje IP adrese i drugih parametara putem HTTP zahteva serveru koji se nalazi na fiksnoj IP adresi (slika 3.8 (a)). Nakon registracije uređaja, dispečer komponenta servera prikuplja TEDS sadržaj dodavanjem aperiodičnih STWS poslova i upisuje taj sadržaj u bazu podataka. Kada se detektuju svi prisutni pretvarači i prikupe podaci o njihovim karakteristikama, operater pomoću Web menadžera konfiguriše mrežu i startuje razmenu podataka, što je prikazano korakom (b).

Dispečer je realizovan tako da procesira listu vremenski sortiranih poslova, pri čemu se sve operacije izvršavaju sekvencijalno, kao na slici 3.8 (c). Dispečer inicira sve upise i čitanja podataka u mreži, što doprinosi stabilnosti sistema, sa obzirom na to da su eliminisani potencijalni zahtevi pametnih pretvarača i aplikacija, koji u slučaju da se šalju serveru suviše često mogu da ugroze njegov rad. U opisanom modelu, server prima zahteve samo prilikom registracije uredjaja, dok se prilikom razmene podataka izmedju prijavljenih uredjaja koristi klijentski STWS interfejs. Time se zapravo postiže da server veći

deo vremena radi kao klijent koji kontroliše komunikaciju, što nije omogućeno u slučaju tipične realizacije servera koja podrazumeva samo obradu dolazećih zahteva.



Slika 3.8 Postupak konfiguracije sistema putem centralnog servera: (a) Prijavljivanje, otkrivanje uređaja i učitavanje TEDS sadržaja; (b) Povezivanje otkrivenih kanala putem GUI-a; (c) Regularan rad sistema koji podrazumeva izvršavanje periodičnih *read/write* poslova.

Realizacija servera će u ovoj tezi biti zasnovana na korišćenju Java tehnologija, mada se prema datom modelu servera mogu koristiti i druge tehnologije. Korišćenjem objektno orijentisanih programskih jezika, lako se može postići uniforman pristup različitim poslovima zahvaljujući mehanizmu polimorfizma, što olakšava implementaciju dispečera.

3.4. Verifikacija modela

Novi model se zasniva uvođenju namenskog pretvaračkog modula sa integrisanim STWS serverom, virtuelnog pretvaračkog modula i centralnog menadžerskog čvora. Prvobitno će biti data verifikacija modela pretvaračkog modula putem studije slučaja osmatranja parametara okoline u klasičnom

klijent-server modelu. Zatim će biti prikazan primer osmatranja parametara okoline prema novom modelu sa centralnim serverom i aktivnom komponentom u formi dispečera.

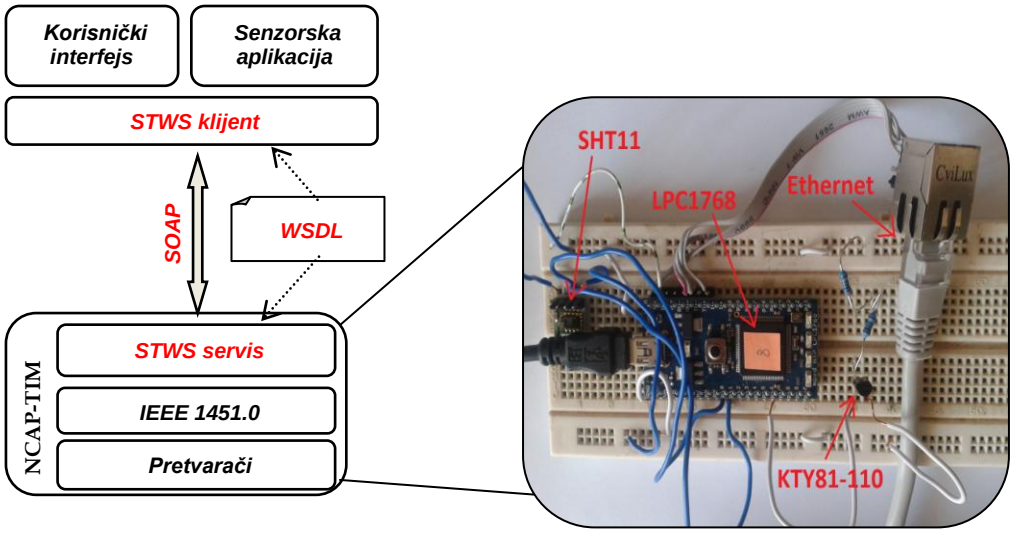
3.4.1. Upotreba pametnog pretvarača na bazi ARM mikrokontrolera

U okviru studije slučaja osmatranja parametara okoline [26], podaci se transportuju od sloja fizičkih uređaja u vidu mernih vrednosti, pa do senzorskih aplikacija višeg nivoa koje vrše procesiranje datih podataka i prikaz rezultata. STWS servisi se mogu smatrati posredničkim slojem koji olakšava opisani prenos podataka time što apstrakuje detalje komunikacije sa fizičkim uređajem.

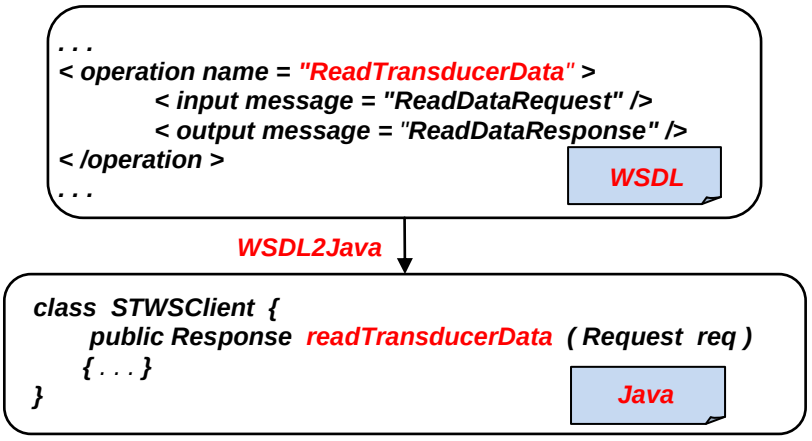
Prvobitno, generički modul pametnog pretvarača na bazi ARM mikrokontrolera, opisan u sekciji 3.1, povezan je sa temperaturnim senzorom KTY81-110 [27] koji je korišćen za merenje temperature vazduha (slika 3.9). Takođe, za potrebe testiranja, povezan je i SHT11 [28] senzor za merenje temperature i relativne vlažnosti vazduha. Očitavanja senzora se vrše periodično, na svake dve sekunde, a šalju se klijentskoj aplikaciji po pozivu servisa za čitanje podataka. Po svakom očitavanju izvršava se algoritam za izračunavanje tačke rose, pri čemu rezultat algoritma zajedno sa mernim vrednostima čini set podataka za slanje klijentskoj aplikaciji. Primer aplikacije u kojoj je, pored drugih parametara, praćenje tačke rose bitno je detekcija leda na saobraćajnim putevima.

Za realizaciju klijentske aplikacije korišćeno je NetBeans razvojno okruženje i Java. Na osnovu postojećeg WSDL fajla, automatski su generisane klijentske funkcije za pozivanje implementiranih servisa. Slika 3.10 ilustruje deo XML specifikacije koji definiše operaciju servisa za čitanje podataka i odgovarajući generisani Java kod klijenta. Argument koji prihvata Java metoda *readTransducerData* je objekat koji sadrži parametre poziva, na osnovu kojih se generiše tekstualna SOAP/XML poruka zahteva spremna za slanje serveru. Aplikacija je dodatno opremljena bazom podataka za smeštanje mernih i drugih

podataka senzora, kao i HTML interfejsom pomoću koga korisnik može da inicira operacije otkrivanja senzora i njihovo periodično očitavanje.



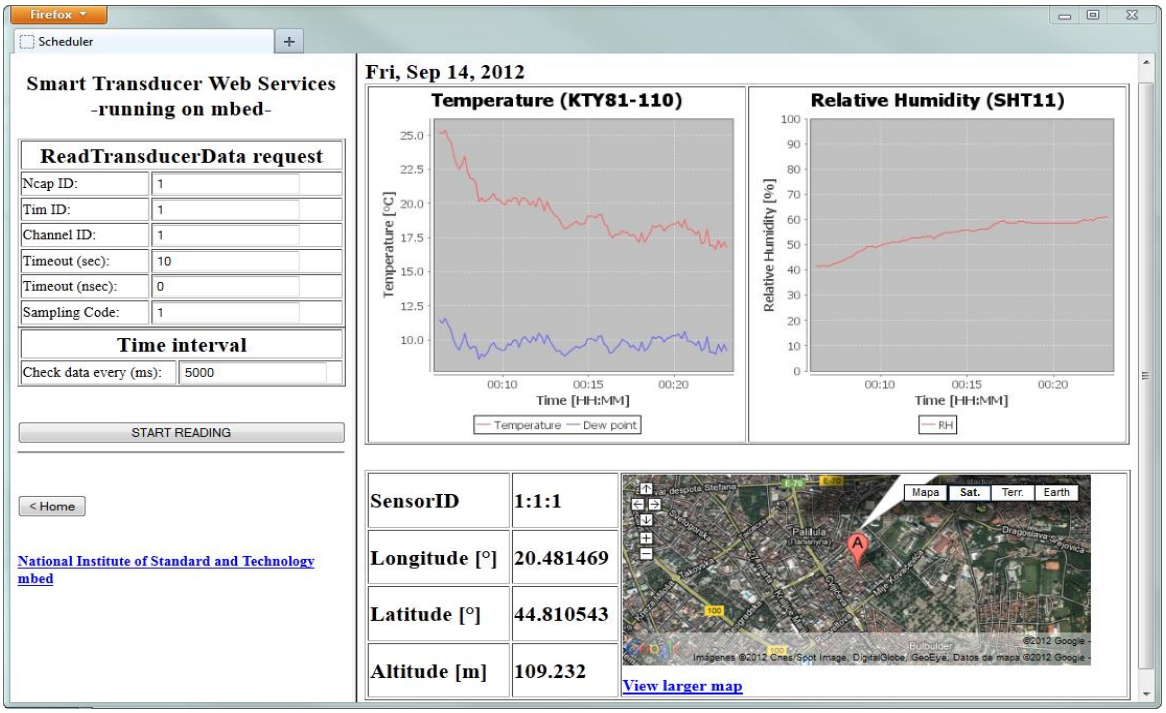
Slika 3.9 Pametni pretvarač na bazi ARM mikrokontrolera za merenje temperature i relativne vlažnosti vazduha.



Slika 3.10 Generisanje klijentske Java metode na osnovu WSDL definicije.

Sa obzirom na to da su putem servisa dostupni i opisni podaci uređaja, moguće je koristiti isti klijentski interfejs za različite NCAP/TIM module. Pri tome je potrebno da realizator uređaja unapred odredi sadržaje TEDS fajlova, kako bi klijentska aplikacija mogla putem poziva servisa da ustanovi o kom tipu senzora je reč i kakve su njegove karakteristike.

Prvobitno, klijentska aplikacija poziva servise za otkrivanje senzora i smešta u bazu podatke o dostupnim senzorima kao i dodatne TEDS podatke poput geografskih koordinata, opisa TIM modula i slično. Zatim korisnik bira vremenski interval za periodično pozivanje servisa za očitavanje senzora, koji je u testu postavljen na 5 sekundi. Posle svakog poziva, ažuriraju se baza u koju se smeštaju merne vrednosti i vrednosti tačke rose, kao i displej za prikaz promene tih vrednosti u vremenu. Na displeju za temperaturu na slici 3.11 se može uočiti temperatura vazduha koju obezbeđuje pametni senzor (crvena boja) i sračunata tačka rose (plava boja). Podaci o lokaciji senzora se prikazuju na displeju na osnovu geo-lokacijskih TEDS sačuvanih u bazi podataka.



Slika 3.11 Displej za prikaz mernih podataka i lokacije senzora.

Rešenje predloženo u okviru studije slučaja daje osnovu za razvoj distribuiranih senzorskih aplikacija. Pristup podacima senzora vrši se metodama klijentske aplikacije koje apstrakuju pozive udaljenih servisa. Pomenute metode mogu biti generisane u različitim programskim jezicima na osnovu početne WSDL specifikacije. U ovom radu je dat primer pristupa

senzorima temperature i relativne vlažnosti, ali je model univerzalan po pitanju tipa senzora i senzorske aplikacije.

3.4.2. Detekcija leda na osnovu modela sa centralnim serverom

U nastavku će biti demonstrirana komunikacija i prikazani rezultati u sistemu za detekciju leda. Korišćeni VTIM algoritam daje procenu uslova za formiranje leda, obradom podataka sa senzora temperature vazduha, temperature površine, relativne vlažnosti vazduha i sračunate tačke rose, pri čemu se generiše signal upozorenja na izlazu. Za računanje tačke rose DP koristi se uobičajena aproksimacija Magnus formule:

$$DP = \frac{B_1\left(\ln\frac{RH}{100} + \frac{A_1 T_a}{T_a + B_1}\right)}{A_1 - \left(\ln\frac{RH}{100} + \frac{A_1 T_a}{T_a + B_1}\right)} \qquad (3.1),$$

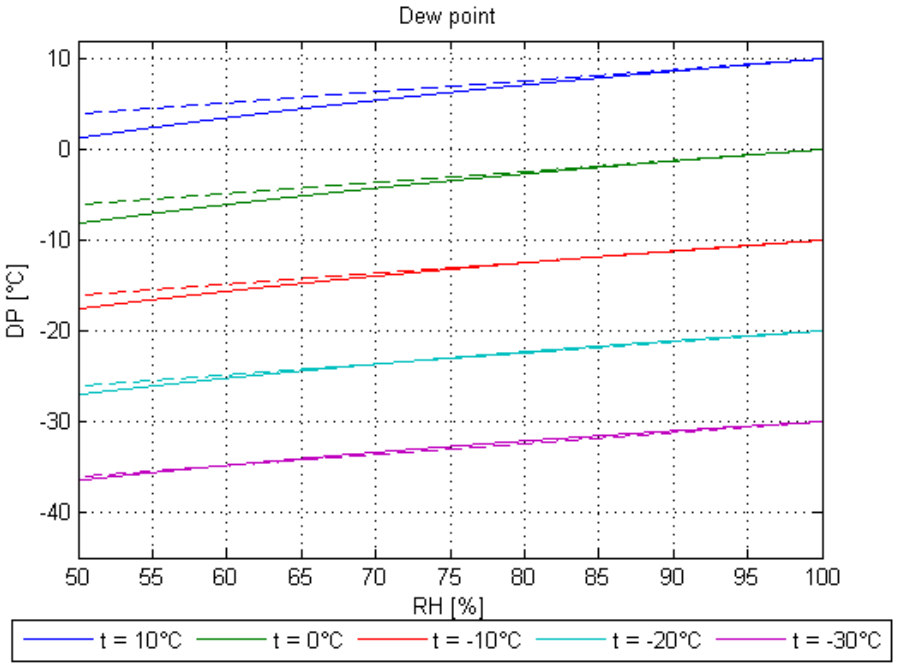
sa vrednostima koeficijenata B_1=273.86, A_1=22.587, gde je RH relativna vlažnost, a T_a temperatura vazduha. Formula (3.1) zahteva intezivno izračunavanje zbog prisustva logaritama, što ne predstavlja poseban problem na 32-bitnom ARM mikrokontroleru koji je korišćen. Ipak, ovde se koristi uprošćena formula koja se dobija linearizacijom izraza (3.1):

$$DP = a(RH - 100) + T_a \qquad (3.2)$$

gde je a=0.1212. U slučaju gde nije moguće koristiti ovu formulu, može se na jednostavan način korigovati greška ili, u krajnjem slučaju, zadržati polazna formula. Na slici 3.12 je dat uporedni prikaz tačke rose dobijen pomoću formula (3.1) i (3.2), za različite vrednosti temperature vazduha.

Na osnovu date formule za tačku rose (3.2), na VTIM mrežnom čvoru se izvršava algoritam za detekciju leda prikazan slikom 3.13. U interesu je detektovati vrednosti temperature ispod 0°C ili bliske ovoj vrednosti. Osim toga, uočavamo da postoje dva slučaja u kojima dolazi do otpuštanja vlage iz

vazduha: kada je temperatura podloge niža od vrednosti tačke rose i kada je temperatura vazduha niža od vrednosti tačke rose.



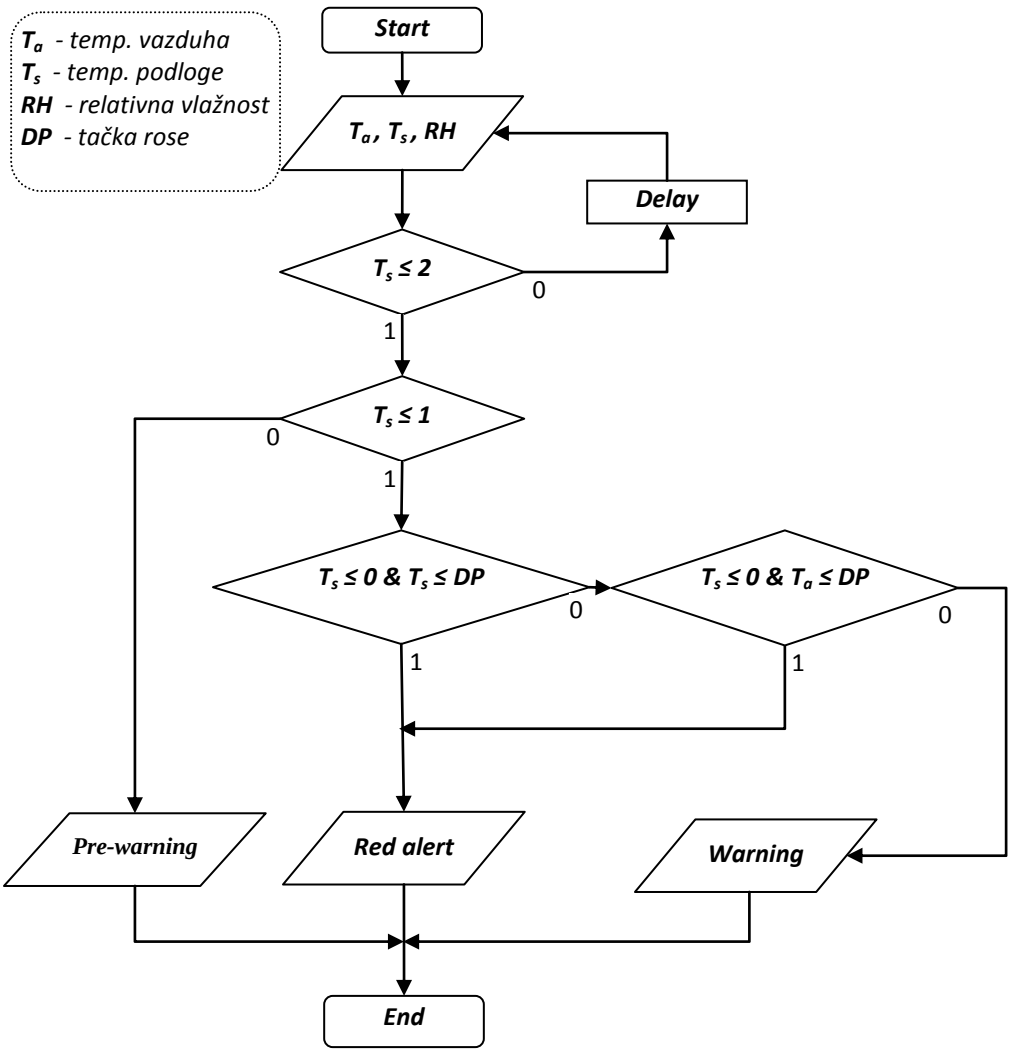
Slika 3.12 Uporedni prikaz vrednosti tačke rose dobijenih pomoću Magnus formule (-) i aproksimacije(--), gde je *t* temperatura vazduha.

Pretpostavljeno je da nema potrebe računati tačku rose za temperature podloge iznad 1°C. Stanje sa dijagrama označeno sa *Red Alert* označava postojanje uslova za pojavu poledice na posmatranoj podlozi.

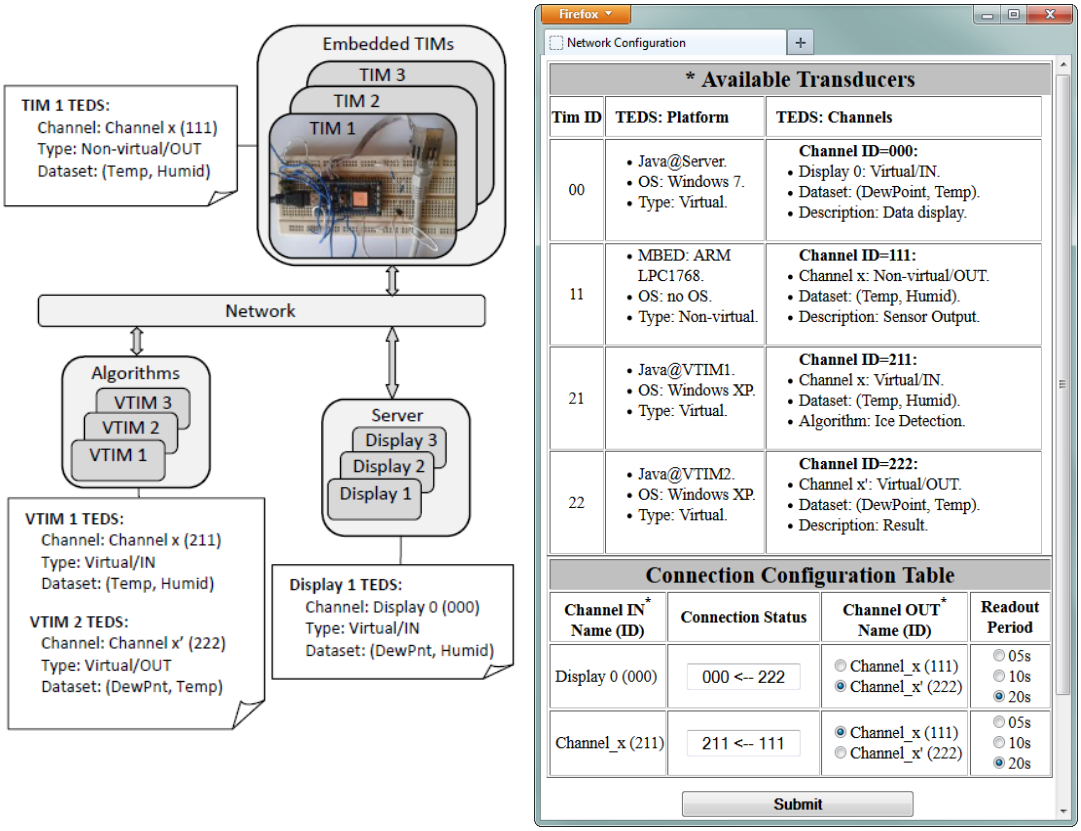
Na slici 3.14 (a) je prikazana mreža koja se koristi u okviru eksperimenta sa TEDS podacima pojedinih čvorova, gde je displej za prikaz rezultata, radi jednostavnosti, ugrađen u serversku aplikaciju. Temperaturni senzor i senzor relativne vlažnosti (TIM 1: Channel x (111) sa slike), su povezani sa mikrokontrolerom slično kao i ranije.

Pritom, VTIM 1 obezbeđuje ulaz datog algoritma preko kanala Channel x (211), dok se rezultat algoritma upisuje u kanal Channel x' (222) koji pripada komponenti VTIM 2. Web displeju koji je registrovan na serveru kao specijalizovani VTIM, pristupa se preko kanala Display 0 (000).

Sadržaj konfiguracione stranice nakon registracije uređaja je prikazan slikom 3.14 (b). Gornja tabela prikazuje TEDS podatke kako bi operatoru bili dostupni opisi dostupnih uređaja, na osnovu čega bira način povezivanja. U datom slučaju, kanali fizičkih senzora se povezuju sa ulaznim VTIM kanalima, koji se vide kao virtuelni aktuatori, odnosno služe kao ulaz u algoritam. Pritiskom na **Submit** dugme, podaci konfiguracije se upisuju u bazu podataka i startuje se periodična razmena podataka duž definisanih putanja dodavanjem *read/write* poslova u listu dispečera.



Slika 3.13 VTIM algoritam za detekciju uslova pojave leda.

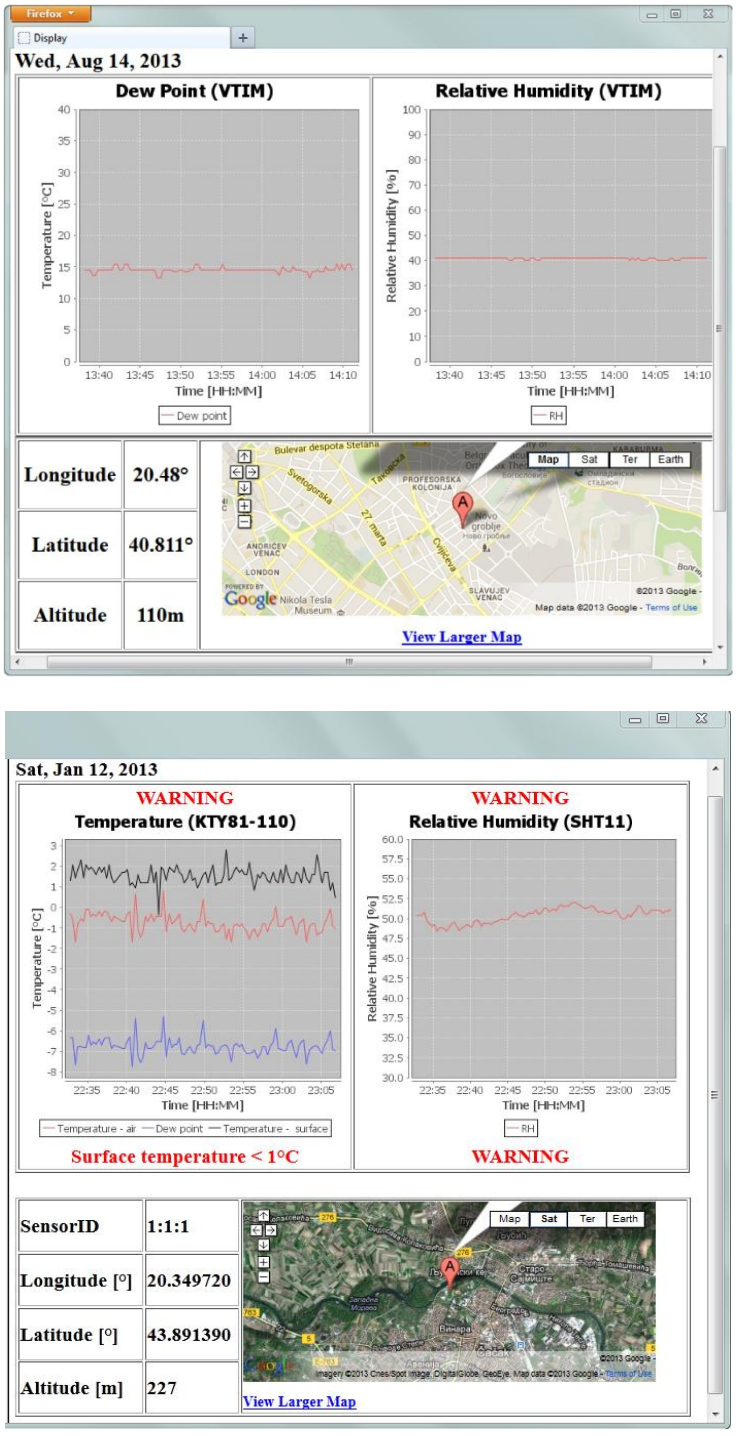


(a) **(b)**

Slika 3.14 Prikaz studije slučaja za verifikaciju modela mreže sa centralnim menadžerskim čvorom: (a) Test postavka; (b) Stranica za konfiguraciju komunikacije.

Displej koji je povezan na izlaz algoritma koji obrađuje podatke dobijene u letnjem periodu je prikazan slikom 3.15 (gore). Tada, geo-lokacijske informacije prikazane za dati kanal se odnose na lokaciju fizičkog senzora, a ne na lokaciju VTIM-a. Takođe, slikom 3.15 (dole) se prikazuje rezultat izvršavanja algoritma za merne podatke dobijene u zimskom periodu (temperatura vazduha: crvena boja; tačka rose: plava; temperatura podloge: crna). Na prikazanom primeru je verifikovan novi model implementacije koji omogućava platformski nezavisnu dislokaciju funkcionalnosti u okviru mreže što vodi ka pouzdanijoj realizaciji namenskog pametnog pretvarača i optimizaciji njegove potrošnje. Nova arhitektura je primenljiva u slučaju realizacije skalabilnih

distribuiranih sistema sa uniformnom integracijom pametnih pretvarača i njima pridruženih aplikacija.



Slika 3.15 Rezultat izvršavanja algoritma za detekciju leda.

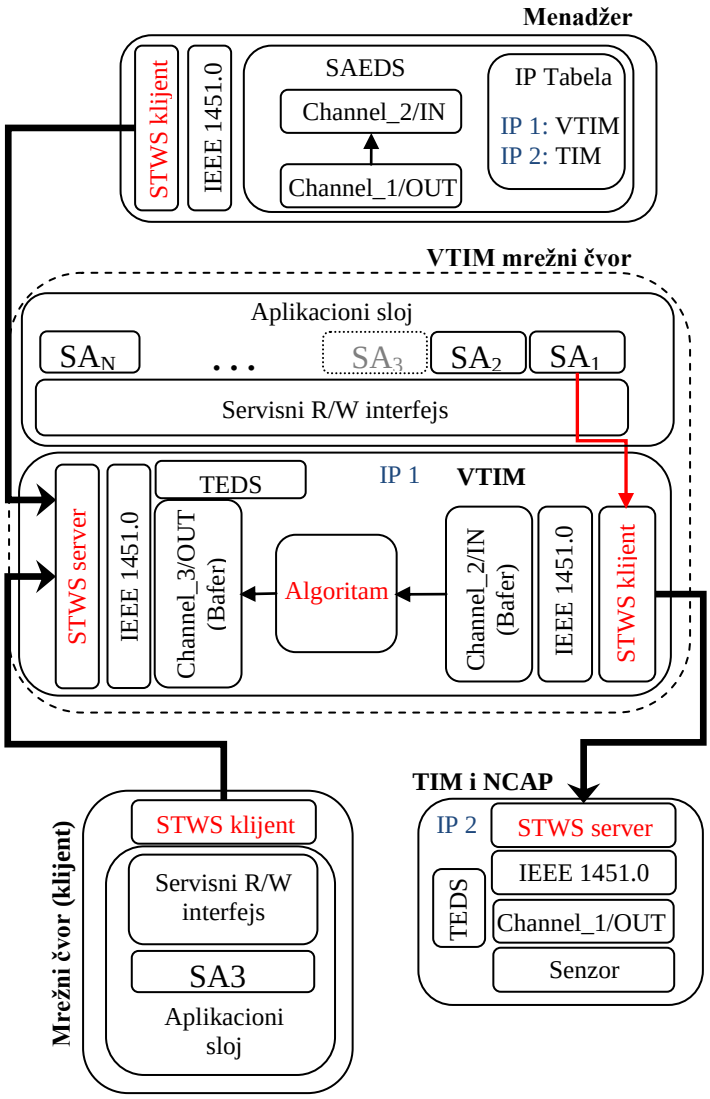
4. SERVISNI AGENTI KAO AKTIVNE KOMPONENTE U MREŽAMA PAMETNIH PRETVARAČA

Prethodno prikazani model daje rešenje menadžmenta mreže i uvodi aktivnu komponentu u formi dispečera za potrebe automatizovane razmene podataka. Međutim, centralni server kao stalno prisutni posrednik u komunikaciji uvodi dodatno kašnjenje na komunikacionoj putanji, pri čemu u prikazanim primerima TIM može da šalje podatke direktno VTIM modulu bez ijednog posrednika. Sa druge strane, usvajanje SOA [29-32] principa i upotreba agenata [33-37] u industrijskim kontrolnim sistemima su priznati kao perspektivna opcija za mrežnu integraciju heterogenih uređaja i sistema. Dati model je unapređen uvođenjem servisnih agenata u formi aktivnih komponenti, koji se mogu nalaziti bilo gde u mreži i obezbediti direktnu komunikaciju mrežnih nodova, čime se obezbeđuje proizvoljna mrežna topologija. U ovom poglavlju, prikazan je model koji se dobija uvođenjem servisnih agenata, kao i integracija servisa za predikciju signala i model integracije prediktivnog kontrolnog algoritma u jednom takvom modelu.

4.1. Opšta mrežna arhitektura

U slučaju da više mrežnih nodova komunicira preko centralnog servera, može doći do zasićenja prilikom procesiranja vremenske liste poslova od strane dispečerske komponente. I pored toga što se kašnjenje tokom procesiranja liste može ublažiti uvođenjem neblokirajućih *read/write* poziva udaljenih servisa, jasno je da se efikasnija realizacija dobija eliminisanjem posredničke komponente sa putanje podataka.

Nova arhitektura zadržava podelu komponenti u sistemu na aktivne i pasivne, sa tom razlikom što se uvodi model koji je orijentisan ka realizaciji putanje podataka, gde centralni čvor i dalje učestvuje u konfiguraciji putanje i drugim podešavanjima [38]. Opšti model mreže je prikazan slikom 4.1.



Slika 4.1 SOA model orijentisan ka putanjama podataka, sa servisnim agentima kao aktivnim komponentama.

Kao i ranije, na osnovu IEEE 1451.0 standarda, pretvarači su integrisani u sistem putem pretvaračkog modula TIM. Na sličan način interfejsni modul za entitete koji nisu pretvarači, poput algoritama za obradu podataka i ulazno/izlaznih uređaja, dat je u formi VTIM-a. Na slici je prikazan opšti algoritam u formi VTIM-a koji razmenjuje podatke sa menadžerom, TIM-om i klijentom. VTIM i TIM čvor komuniciraju direktno, bez posrednika, dok je uloga menadžera da obezbedi parametre takve komunikacije kroz posebno definisane TEDS. Dopremanje TIM senzorskih podataka do algoritma, kao i

čitanje rezultata obrade algoritma, vrše se periodičnim izvršavanjem aktivnih komponenti SA_i u okviru aplikacionog sloja. SA_i izoluje detalje poziva servisa, bez obzira na to da li su lokalni ili udaljeni, od ostatka aplikacije i omogućava razmenu podataka i procesiranje duž definisane putanje podataka. Ovakva komponenta radi sa predefinisanom konfiguracijom u zavisnosti od potreba za brzinom obrade i razmene podataka.

STWS klijent na strani VTIM čvora, povlači paket mernih podataka iz TIM-a sa predefinisanim periodom, tako da je ova akcija registrovana u modelu kao SA_1. Algoritam obrađuje merne podatke dobijene od strane TIM modula, pri čemu se algoritam startuje po samom pristizanju podataka. Rezultat obrade se smešta u poseban izlazni bafer (Channel 3) kako bi bio dostupan klijentskoj aplikaciji. SA_3 komponenta je dislocirana na stranu klijenta i služi za periodično očitavanje rezultata obrade algoritma.

VTIM čvor sadrži TEDS meta-podatke definisane IEEE 1451.0 standardom, ali i podatke koji se koriste u procesu konfiguracije: SAEDS (Service Agent Electronic Data Specification) i TEDS čiji sadržaj zavisi od prirode samog algoritma. Nadogradnjom plug & play modela zasnovanog na TEDS sadržaju, uvode se nove SAEDS komponente za specifikaciju/konfiguraciju jedinstvenih putanja podataka, što će detaljnije biti objašnjeno kasnije u tezi. STWS server VTIM-a prihvata konfiguracione podatke u SAEDS formatu, kako bi kasnije SA znao odgovarajuću IP adresu i broj ciljanog pretvaračkog kanala kome pristupa. Takođe, STWS server prihvata zahtev *read* servisa koji šalje klijentski mrežni čvor.

4.2. Verifikacija modela baziranog na upotrebi servisnih agenata

Arhitektura distribuiranog sistema, algoritmi za kontrolu procesa, komunikacioni protokoli i formati podataka i imunost na probleme mrežne komunikacije su neke od tema koje su detaljno razmatrane u radovima [39-43].

U okviru verifikacije modela, biće prikazana mreža sa algoritmom za predikciju proizvoljnog signala na bazi neuralne mreže, kao i mreža sa prediktivnim algoritmom za upravljanje konkretnim fizičkim procesom, pri

čemu se u oba slučaja algoritmi realizuju u formi VTIM pasivne komponente. Predikcija budućih odbiraka signala, kako je pokazano u narednim sekcijama, predstavlja pogodno rešenje za prevazilaženje slučajnog mrežnog kašnjenja i gubitka paketa.

4.2.1. Integracija servisa za predikciju signala

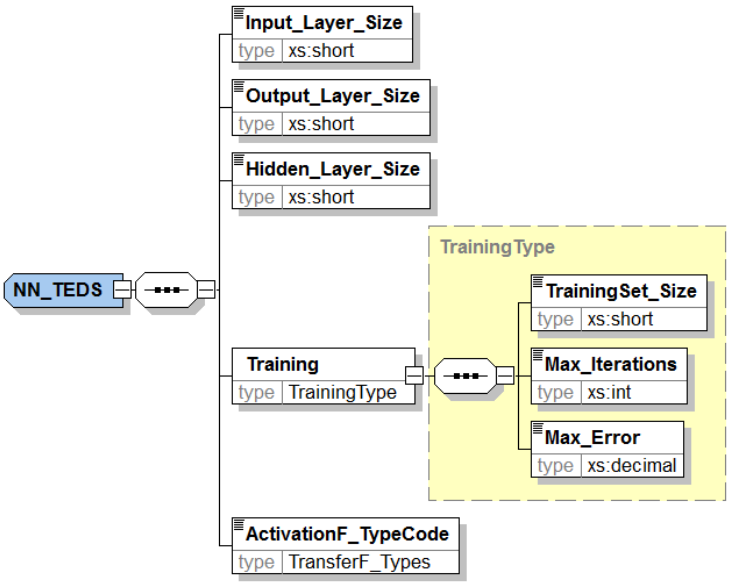
Distribuirani kontrolni sistemi se mogu realizovati na bazi postojeće mrežne infrastrukture popute Ethernet-a. Međutim, u radu sa takvim mrežama, javlja se problem gubitaka paketa i nepredvidljivog mrežnog kašnjenja [39]. Pomenuti problemi nisu adresirani ni putem IEEE 1451 specifikacije, ni putem datog SOA rešenja, pa je zbog toga neophodno definisati dodatne sistemske komponente za potrebe realizacije merno-kontrolnog sistema.

U primeru koji sledi, prema opštoj arhitekturi sa slike 4.1, kao algoritam se koristi prediktivni servis sa svrhom generisanja budućih odbiraka proizvoljnog signala, čime se kompenzuje mrežno kašnjenje. Tada, vremenski prozor predikcije mora biti dovoljno širok da obuhvati vremenski trenutak pristizanja paketa sa budućim odbircima na željenu destinaciju. Zatim, odredišni čvor procesira paket i upoređuje vremenske reference pristiglih odbiraka sa lokalnom vremenskom referencom kako bi izdvojio odgovarajuće podatke i prosledio ih izlazu aktuatora (TIM) ili ulazu algoritma (VTIM). Lokalna referenca se realizuje u formi brojača sa rezolucijom 1ms, koji poseduje svaki mrežni čvor, a koji se sinhronizuje putem posebnog STWS servisa za sinhronizaciju. STWS servis za sinhronizaciju se izvršava povremeno, tako što menadžerski čvor emituje sinhronizacione poruke putem multicast mehanizma.

Servis za predikciju signala je implementiran u formi neuralne mreže, upotrebom Java biblioteke [44]. Korišćenjem razvojnog okruženja NetBeans IDE i date biblioteke, vrši se efikasna integracija prediktivnog algoritma u formi VTIM-a u IEEE 1451.0 mrežu pametnih pretvarača. Servisni agent prosleđuje podatke VTIM algoritmu, kao pasivnoj komponenti, putem FIFO bafera, dok se

rezultat izvršavanja algoritma smešta u bafer kako bi bio dostupan klijentskom čvoru putem standardnog Web servis interfejsa.

Konfiguracija prediktivnog algoritma se ostvaruje na pogodan način upotrebom TEDS komponenti u posebnom formatu. Na slici 4.2 je prikazan bazičan format za konfiguraciju servisa za predikciju signala putem neuralne mreže.

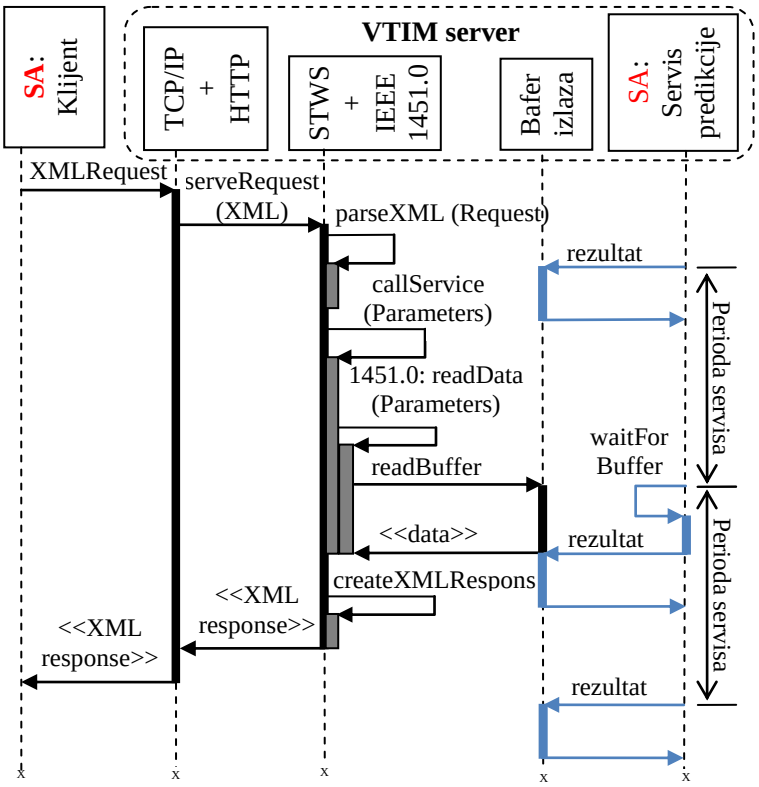


Slika 4.2 Format aplikacione TEDS komponente za parametrizaciju servisa za predikciju signala baziranog na neuralnoj mreži.

Radi jednostavnosti, pretpostavlja se da neuralna mreža ima samo jedan skriveni sloj neurona koji može biti parametrizovan. Pri svakom očitavanju senzora, merni podaci se dodaju u bafer istorije merenja. Tada se koristi parametar **TrainingSet_Size** kako bi se odredilo koliko podataka se uzima iz bafera za kreiranje skupa podataka za obučavanje neuralne mreže. Ostali parametri kofigurišu broj neurona, maksimalan broj iteracija prilikom obučavanja, maksimalnu grešku obučavanja i tip aktivacione funkcije.

Komunikaciona sekvenca između klijentske aplikacije i servisa za predikciju, prikazana je slikom 4.3. SA na klijentskoj strani šalje SOAP/XML zahtev VTIM serveru, koji parsira zahtev i prosleđuje dobijene parametre

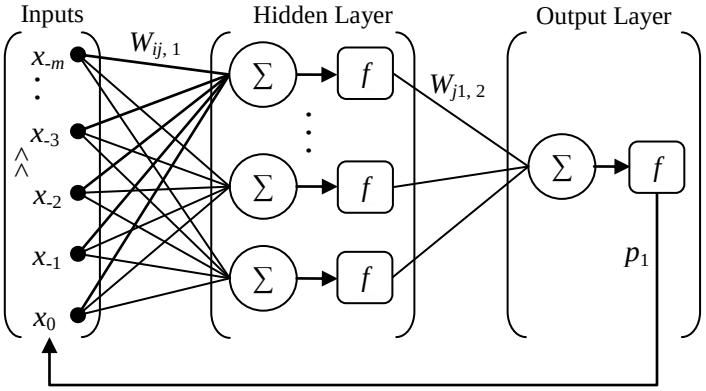
readData operaciji IEEE 1451.0 sloja. Nakon toga, poslednji rezultat algoritma čije je izvršavanje inicirano od strane zasebnog SA, se očitava i vraća klijentu. U slučaju da se zahteva očitavanja TEDS podataka od strane menadžerskog čvora, nakon parsiranja XML zahteva, poziva se *readTeds* metoda u okviru IEEE 1451.0 sloja.



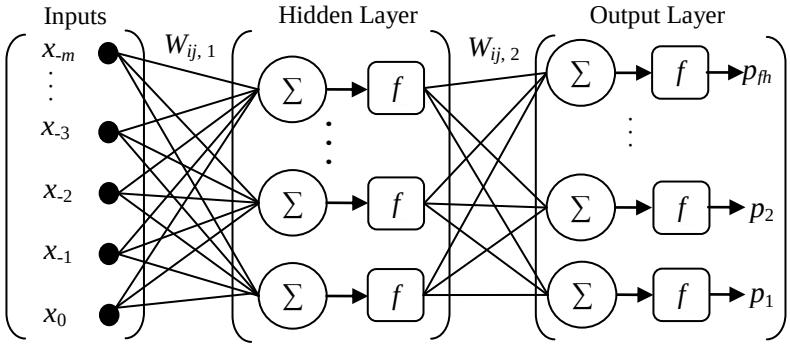
Slika 4.3 Sekvenca komunikacije prilikom pristupa klijentske aplikacije servisu za predikciju signala.

Kako bi se obezbedila podrška za rad sistema u realnom vremenu, u algoritmima distribuiranog merenja, procesiranja i prikaza rezultata, neophodno je uvesti vremensku referencu u set podataka. Svaki odbirak signala, bez obzira na to da li je rezultat merenja ili predikcije, ima odgovarajući podatak o vremenu. Predikcioni servis zapravo dopunjava set podataka budućim vrednostima i odgovarajućim vremenskim referencama. Prema tome, glavni cilj predikcionog servisa jeste kompenzacija velikog vremenskog kašnjenja koje se javlja pri komunikaciji putem STWS servisa.

Rezultati testiranja su dati za dva tipa neuralne mreže koji su prikazani slikom 4.4, gde je $W_{ij,\,k}$ težinski koeficijenti *i*-te ulazne sinaptičke veze neurona *j* u sloju *k*. Aktivaciona funkcija neurona *f* je hiperbolički tangens. Ulazi x_0, x_{-1}, ..., x_{-m} su najnoviji merni odbirci dobijeni iz FIFO bafera koji sadrži vremensku sekvencu podataka senzora. Veličina bafera je veća od broja odbiraka na ulazu neuralne mreže koji iznosi m+1. Konfiguracija neuralne mreže putem menadžerskog noda omogućava podešavanje veličine neuralnih slojeva, na osnovu TEDS formata posebno prilagođenog aplikaciji (slika 4.2).



(a)

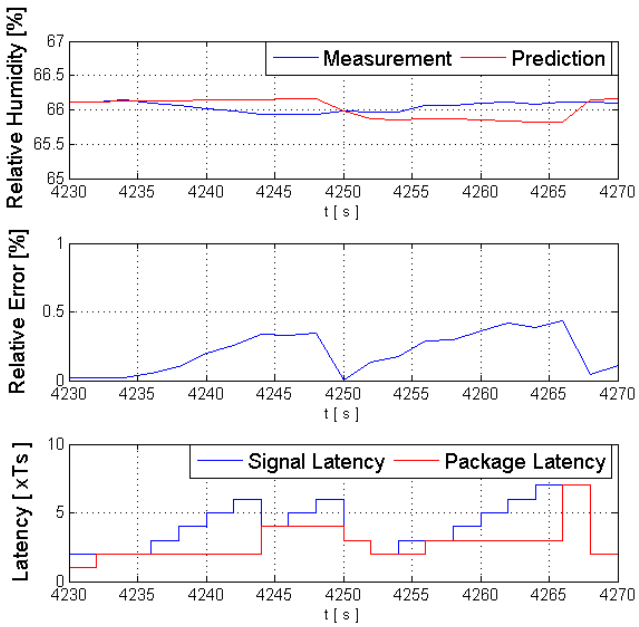


(b)

Slika 4.4 Test konfiguracije: (a) Feedback neuralna mreža, za iterativno sračunavanje budućih odbiraka; (b) Feed-forward neuralna mreža za istovremeno sračunavanje budućih odbiraka.

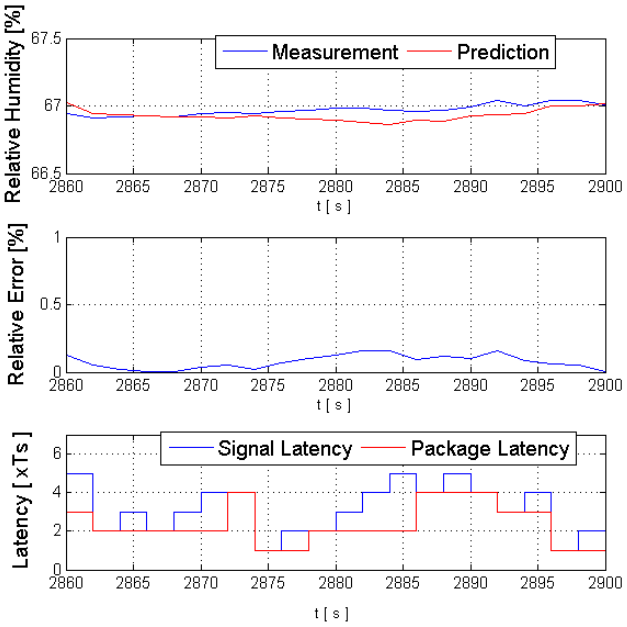
Slučaj pod (a) podrazumeva da prvi budući odbirak (p_1) zavisi samo od mernih podataka, a da svaki sledeći odbirak zavisi i od mernih podataka i od predikcijom dobijenih podataka, što znači da se računanje vrši iterativno za svaki novi odbirak predikcije, gde broj iteracija zavisi od horizonta predikcije *fh*. U slučaju pod (b), svi budući odbirci (p_1, p_2, ..., p_{fh}) se procenjuju paralelno i pretpostavlja se da zavise samo od mernih podataka. U tom slučaju, broj neurona u izlaznom sloju zavisi od horizonta predikcije *fh*.

U test scenariju, koristi se senzor relativne vlažnosti vazduha, sa periodom odabiranja T_s=2s, i analizira se rad sistema sa slučajnim mrežnim kašnjenjem koje varira u intervalu od T_s do 10xT_s (2-20s). Relevantni talasni oblici signala na putanji podataka, za prethodno definisane neuralne mreže date slikama 4.4 (a) i 4.4 (b), dati su slikama 4.5 i 4.6, respektivno.



Slika 4.5 Predikcija signala relativne vlažnosti u slučaju upotrebe feedback neuralne mreže, na vremenskom intervalu od 40s: **(gore)** relativna vlažnost (*Measurement*) i predikcija (*Prediction*); **(sredina)** greška predikcije (*Relative Error*); **(dole)** kašnjenje (*Latency*) izraženo kao celobrojni umnožak periode odabiranja T_s.

Na osnovu grafika se može zaključiti da slučaj sa slike 4.6 daje nešto bolje rezultate predikcije za dati sporopromenljivi signal. Predikcioni servis se poziva sa periodom koja je ista kao i perioda odabiranja, a zatim se simulira propagacija paketa sa slučajnim mrežnim kašnjenjem na putanji podataka. Kao posledica, paketi mogu stizati na klijentsku stranu u drugačijem redosledu, pa je zadatak klijentskog čvora da filtrira pakete koji su stariji u odnosnu na one koji se trenutno procesiraju. Zbog toga je verovatnije odbijanje paketa sa velikim mrežnim kašnjenjem. To se može videti i na graficima, gde je kašnjenje paketa uglavnom manje od 5xT_s, iako se razmatra opseg od 10xT_s u okviru simulacije.



Slika 4.6 Predikcija signala relativne vlažnosti u slučaju upotrebe feed-forward neuralne mreže, na vremenskom intervalu od 40s: **(gore)** relativna vlažnost i predikcija; **(sredina)** greška predikcije; **(dole)** kašnjenje izraženo kao celobrojni umnožak periode odabiranja T_s.

Podrška za sinhronizaciju je neophodna u ovom slučaju za korektno filtriranje paketa. Ukoliko novi paket izostaje, aplicira se sledeći odbirak iz

postojećeg paketa. U tom slučaju, kašnjenje signala se inkrementira za T_s, zbog čega se na graficima dobija stepenasti oblik.

Neuralne mreže prikazane na slici 4.4 se mogu upotrebiti za predikciju i drugih senzorskih signala, koji ne predstavljaju relativnu vlažnost. Pritom, uticaj neuralnih prediktora na kvalitet signala zavisi od konkretne aplikacije. U prikazanom slučaju, mreža se koristi kao prediktor vremenske sekvence, koji procenjuje trend signala na osnovu istorije merenja senzora. Prikazana arhitektura omogućava efikasnu integraciju proizvoljne neuralne mreže, tako da se dobija podrška za implementaciju distribuiranih merno-kontrolnih aplikacija. Dodatne opcije mreže, poput obučavanja koje uključuje dnevni trend meteoroloških podataka i slično, mogu se lako dodati naknadno. Kvantitativnu analizu za svaki od ovih slučajeva potrebno je sprovesti zasebno.

I pored toga što je relativna vlažnost vazduha sporopromenljivi signal, data arhitektura podržava i veće učestanosti odabiranja, pri čemu se više mernih odbiraka, zajedno sa vremenskim referencama, smešta u jedan paket za prenos preko servisnog mrežnog interfejsa. Dalje procesiranje i komunikacija se odvijaju slično kao u prikazanom scenariju.

Predložena implementacija sistema daje mogućnost integracije prediktivnog servisa u servisno-orijentisanoj mreži sa IEEE 1451.0 pametnim pretvaračima, gde je minimiziran problem mrežnog kašnjenja i gubitka podataka. Menadžerski čvor pristupa TEDS podacima pametnih pretvarača i dislociranih servisa, kako bi definisao jedinstvene putanje podataka koje su u vremenu regularnog rada sistema pod kontrolom servisnih agenata. Time se potvrđuje fleksibilnost konfiguracije jedinstvenih putanja podataka koja je ustanovljena opštim modelom sa upotrebom servisnih agenata.

Arhitektura mrežnog servisa za predikciju je data u VTIM formi sa nezavisnim ulaznim i izlaznim putanjama. Time se omogućava proizvoljna funkcionalnost servisa koja je u datom slučaju iskorišćena za realizaciju servisa predikcije. Korišćeni format podataka i mehanizam vremenske sinhronizacije daju podršku za integraciju procesa i algoritama sa mogućnošću rada u realnom

vremenu. Prikazani koncept implementacije prediktivnog servisa, pored podrške za rad u realnom vremenu, obezbeđuje interoperabilnost i skalabilnost distribuiranih komponenti u SOA mreži. Detaljnija definicija prikazane arhitekture, sa opisom konfiguracije nezavisnih putanja podataka upotrebom SAEDS-a, kao i detaljima interakcije servisnih agenata i VTIM servisa, zajedno sa studijom slučaja, biće opisana u nastavku.

4.2.2. Distribuirano upravljanje u servisno orijentisanoj mreži pametnih pretvarača

Implementacija kompleksnih kontrolnih sistema, nekada zahteva upotrebu kompleksnih kontrolnih algoritama. Fizički delovi kontrolnog sistema su često dislocirani i zahtevaju mrežnu integraciju i komunikaciju. Upotrebom SOA arhitekture omogućava se migracija funkcionalnosti, zahtevnih po pitanju procesiranja, sa namenske platforme na neku drugu pogodniju platformu u okviru mreže (u VTIM formi).

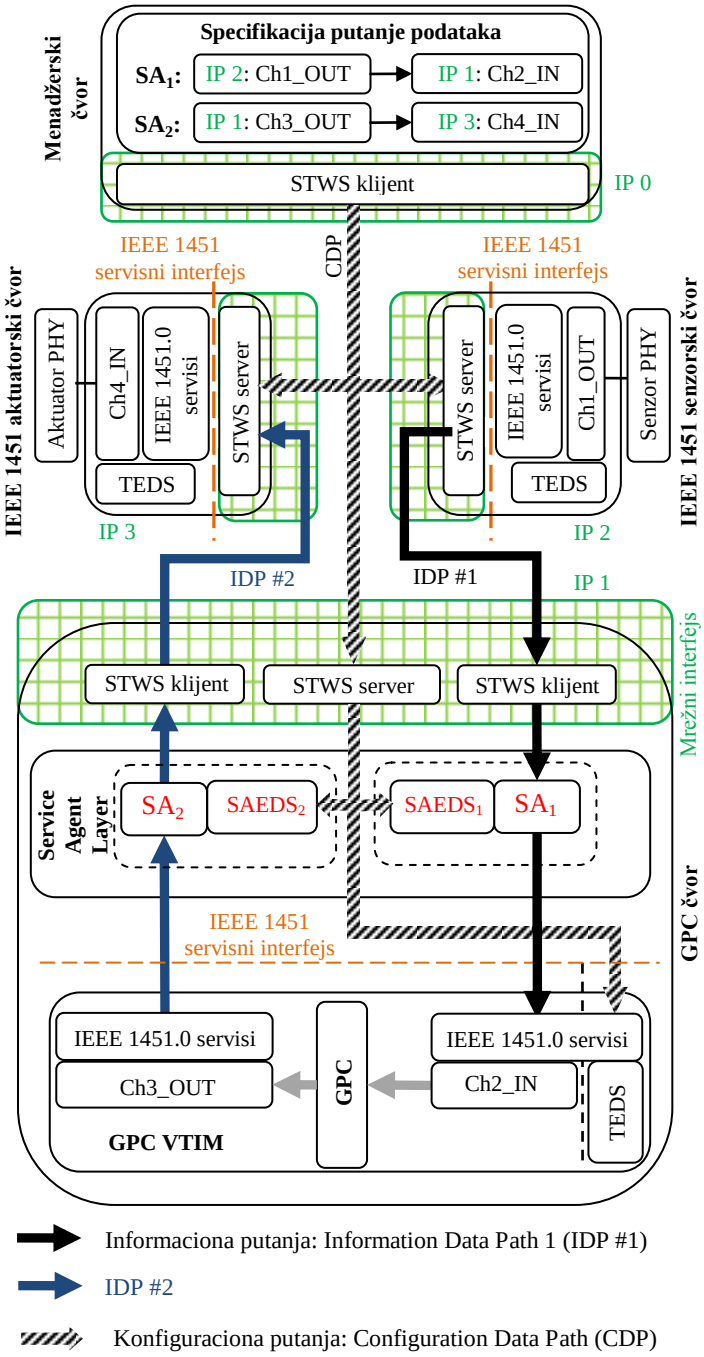
Pošto postojeća mrežna infrastruktura unosi problem slučajnog kašnjenja i gubitka paketa, General Predictive Control (GPC) se nameće kao optimalno rešenje pri izboru kontrolnog algoritma. U tom slučaju, predviđanje budućih vrednosti kontrolnih inkremenata se može iskoristiti za kompenzaciju mrežno-indukovanih problema. Upotreba VTIM komponente omogućava da se GPC kontrolni algoritam dislocira sa namenske platforme na platformu opšte namene. VTIM arhitektura, kao arhitektura pasivne komponente, ne rešava problem razmene podataka sama za sebe, već se ta kontrola uspostavlja uvođenjem aktivnih komponenata u vidu servisnih agenata.

Prethodno opisana neuralna mreža daje predikciju samo na osnovu merne istorije signala, bez upotrebe modela konkretnog fizičkog procesa, što može rezultirati odudaranjem odbiraka dobijenih predikcijom od stvarnih odbiraka potrebnih za kontrolu procesa. Uvođenjem GPC algoritma, predikcija odbiraka se bazira na upotrebi modela konkretnog fizičkog procesa. Kao i ranije, servis za predikciju se može konfigurisati posebno definisanim TEDS. U nastavku će biti dat detaljan opis arhitekture distribuiranog kontrolnog sistema.

Detaljan opis arhitekture distribuiranog kontrolnog sistema, prikazan je slikom 4.7, gde zasebni mrežni čvorovi izvršavaju osnovne funkcije prikupljanja mernih podataka, kontrole i sračunavanja kontrolnih odbiraka, dok je za konfiguraciju i sinhronizaciju zadužen menadžerski čvor. Komunikacija između čvorova u okviru SOA mreže se sprovodi preko STWS interfejsa, kao i ranije, koji je implementiran kao dodatni sloj koji dopunjuje IEEE 1451.0 stek. Predložena arhitektura podrazumeva da svaki fizički mrežni čvor može da sadrži više sistemskih komponenti, između kojih je komunikacija uspostavljena kroz dve odvojene putanje podataka: informacionu putanju (IDP) i konfiguracionu putanju (CDP).

Servisni agent SA, kao ključna komponenta datog distribuiranog sistema, vrši kontrolu toka podataka na IDP putanji i može se implementirati kao nezavisan softverski modul i pokrenuti na bilo kom fizičkom čvoru u mreži. SA pruža podršku za automatizovanu razmenu podataka između različitih pasivnih distribuiranih komponenti (servisa), preko nezavisnih IDP, čime se ukida potreba za mrežnim modelom na bazi zvezdaste topologije. Prema tome, SA obezbeđuje efikasan mrežni transport podataka između dva ili više dislociranih izvorišnih i odredišnih servisa.

Kao što standard IEEE 1451.0 definiše elektronske specifikacije TEDS, a u cilju proširenja plug and play funkcionalnosti na SA domen, koriste se već pomenute SAEDS komponente za specifikaciju/konfiguraciju jedinstvenih IDP putanja. Na taj način, menadžerski čvor vrši uniformnu konfiguraciju svih mrežnih čvorova tako što pristupa njihovim TEDS i SAEDS strukturama podataka preko CDP putanje, upotrebom STWS interfejsa. Menadžerski čvor putem GUI-a omogućava konfiguraciju nezavisnih IDP putanja, kroz povezivanje odgovarajućeg SA sa izvorišnim i odredišnim kanalima, gde kanal predstavlja logičku apstrakciju toka podataka na fizičkom ulazu ili izlazu pametnog pretvarača. Menadžer ima opciju čuvanja odgovarajuće IDP konfiguracije upisom SAEDS podataka servisnog agenta preko CDP putanje u ciljani mrežni čvor.



Slika 4.7 Osnovna struktura i putanje podataka distribuiranog SOA kontrolnog sistema.

Operacije duž IDP putanje podataka se odnose na transport informacija između distribuiranih komponenti u periodu regularnog rada kontrolnog sistema. Ove operacije su u potpunosti kontrolisane od strane servisnih agenata

koji predstavljaju jedine aktivne komponente u inače pasivnoj servisno orijentisanoj arhitekturi.

SA, kao komponenta aplikacionog sloja Service Agent Layer, izoluje idiosinkrazije poziva pojedinih servisa, kako lokalnih tako i udaljenih, i omogućava razmenu podataka sa drugom distribuiranom komponentom na definisanoj IDP putanji. Ova komponenta sprovodi operacije prema predefinisanoj konfiguraciji, sačuvanoj u SAEDS strukturi podataka.

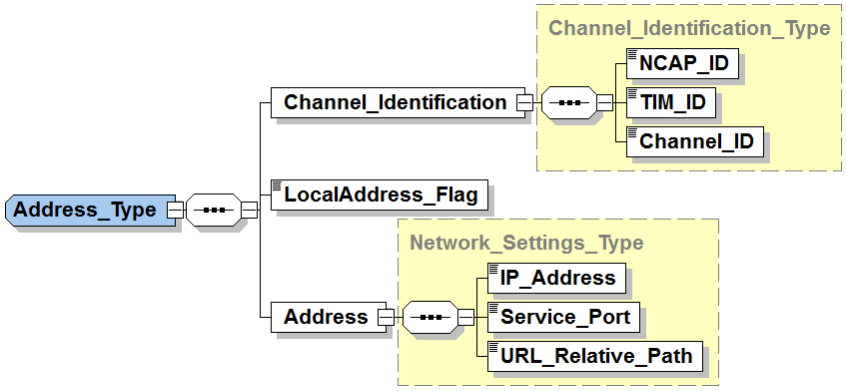
A. Operacije na putanji podataka i menadžment mreže

SA predstavlja posebnu nit izvršavanja programa koja sprovodi periodični upis i čitanje podataka preko nezavisnog lokalnog ili mrežnog servisnog interfejsa. Pritom se koristi softverski ili hardverski tajmer za postizanje sinhronizovanog izvršavanja SA niti. U opštem slučaju, IDP se implementira kao par jedne ili nekoliko operacija čitanja podataka i par ili nekoliko operacija upisa podataka, na odgovarajućim servisnim interfejsima.

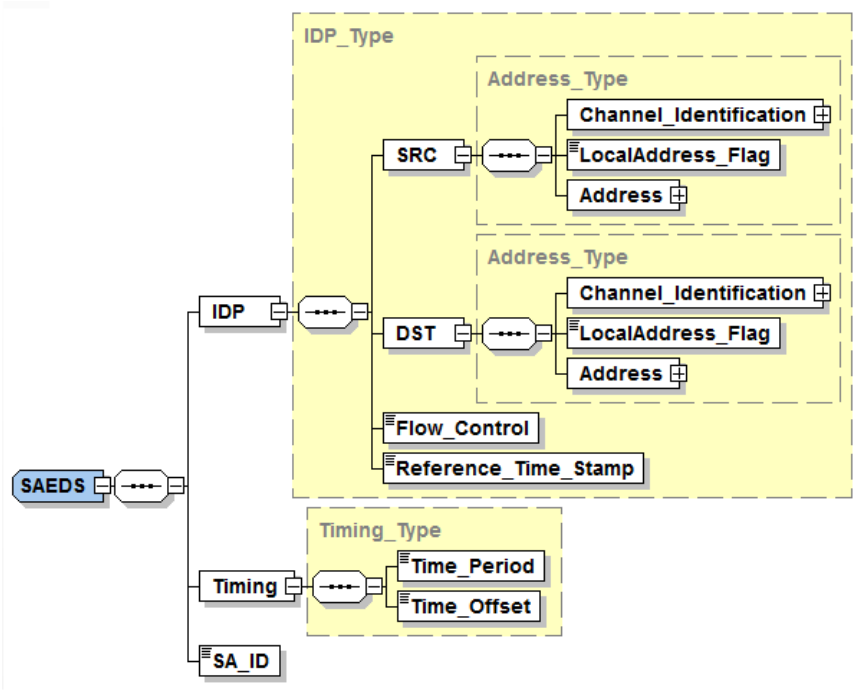
Uvođenjem SAEDS struktura podataka omogućava se postizanje konfiguracije mreže u par jednostavnih koraka: menadžerski čvor prikuplja TEDS podatke registrovanih uređaja, operater definiše SAEDS instance putem GUI interfejsa menadžerskog čvora i šalje ih ka (GPC) VTIM čvoru preko CDP putanje. Nakon završene konfiguracije, startuje se razmena podataka između uređaja na IDP putanji sa nedeterminističkim mrežnim kašnjenjem, što predstavlja motivaciju za uvođenje podrške za vremensku sinhronizaciju operacija na putanji.

Detaljan SAEDS format je prikazan pomoću XML Schema dijagrama sa slike 4.8. U slučaju pod (a), prikazan je kompleksan format za adresiranje kanala koji je dovoljno uopšten da pokrije slučajeve pristupa lokalnom kanalu putem IEEE 1451 servisnog interfejsa i udaljenog pristupa kanalu putem STWS interfejsa. Kanal je logička reprezentacija početne ili krajnje tačke fizičke komunikacije, u zavisnosti od toga da li je kanal ulaznog ili izlaznog tipa. Polje *Channel_ID*, koje pripada kompleksnom polju *Channel_Identification*, predstavlja

jedinstveno dodeljenu identifikaciju svakom od ulaznih i izlaznih kanala u okviru jedne pasivne komponente. Pritom, polja NCAP_ID i TIM_ID se koriste za specifikaciju različitih pasivnih komponenata koje se nalaze iza iste adrese definisane poljem *Address*. *LocalAddress_Flag* se koristi u okviru Service Agent Layer sloja kao indikator poziva servisa preko lokalnog servisnog interfejsa, bez obzira na parametre udaljenog poziva navedene poljem *Address*.



(a)



(b)

Slika 4.8 XML šema za SAEDS: (a) XML šema tip za specifikaciju adrese; (b) SAEDS format za specifikaciju jedinstvene IDP putanje sa vremenskom periodom.

Slika 4.8(b) prikazuje osnovnu SA konfiguraciju koja obuhvata IDP, vremenske i identifikacione parametre. IDP konfiguracija je definisana vezom *SRC* ulaznog i *DST* izlaznog kanala, koja se dobija od strane menadžerskog čvora. Polje *Timing* definiše vremenske parametre koji se odnose na izvršavanje SA, preko *Time_Period* i *Time_Offset* polja.

Kako bi se obezbedila pouzdana isporuka podataka preko IDP, SAEDS šema je proširena sa dva dodatna polja. *Flow_Control* označava da li se od SA komponente zahteva pouzdan transport podataka bez gubitaka informacija u vremenskoj sekvenci podataka, dok se *Reference_Time_Stamp* polje ažurira od strane SA kako bi se sinhronizovao transport informacija na IDP putanji.

Identifikacija svake SAEDS instance kreirane na strani menadžerskog čvora je obezbeđena putem *SA_ID* polja. SAEDS instance se razmenjuju u SOAP/XML formatu između menadžera i SA-ova preko CDP putanje. Predloženi SAEDS format sadrži sve neophodne informacije za konfiguraciju IDP putanje i SA operaciju, dok se dodatni parametri mogu ubaciti u SAEDS kao podrška specifičnim implementacijama.

Kako bi se obezbedila uniformna implementacija SA niti, usvaja se model izvršavanja SA baziran na procesiranju događaja (event-driven), sa obzirom na to da je podržan na različitim namenskim i platformama opšte namene. Mehanizam sinhronizacije se može realizovati upotrebom različitih objekata za komunikaciju i sinhronizaciju niti, i platformski je zavistan. Kada je u pitanju komunikacija, SA nit šalje servisne zahteve preko lokalnog ili STWS interfejsa, pri čemu prima notifikaciju po pristizanju odgovora servisa. Kako bi se istakao model izvršavanja SA baziran na procesiranju događaja, SA nit poziva blokirajuću *wait()* funkciju, kako bi ista prešla u stanje čekanja na notifikaciju povodom tajmerskog događaja ili događaja pristizanja servisnog odgovora u formi poziva funkcije *notify()*.

Nakon notifikacije od strane tajmera, SA izvršavanje podrazumeva operaciju poziva servisa čitanja podataka, prema *SRC* delu specifikacije SAEDS-a. Svaki zahtev servisa čitanja sadrži *SA_ID* identifikaciju, *Transfer_ID*

parametar i vremenski prozor zahtevanih podataka. *Transfer_ID* je lokalno vreme (vremenska referenca; timestamp), koje označava trenutak generisanja zahteva servisa za čitanje podataka i mora biti uključen i u poruku odgovora istog servisa, kao i u naredni zahtev za poziv servisa upisa podataka i njegov odgovor. Parametar *Transfer_ID*, zajedno sa *Reference_Time_Stamp* parametrom, koji su uključeni u zaglavlje paketa, mogu se tretirati na *SRC* strani kao vremenski prozor zahtevanih podataka.

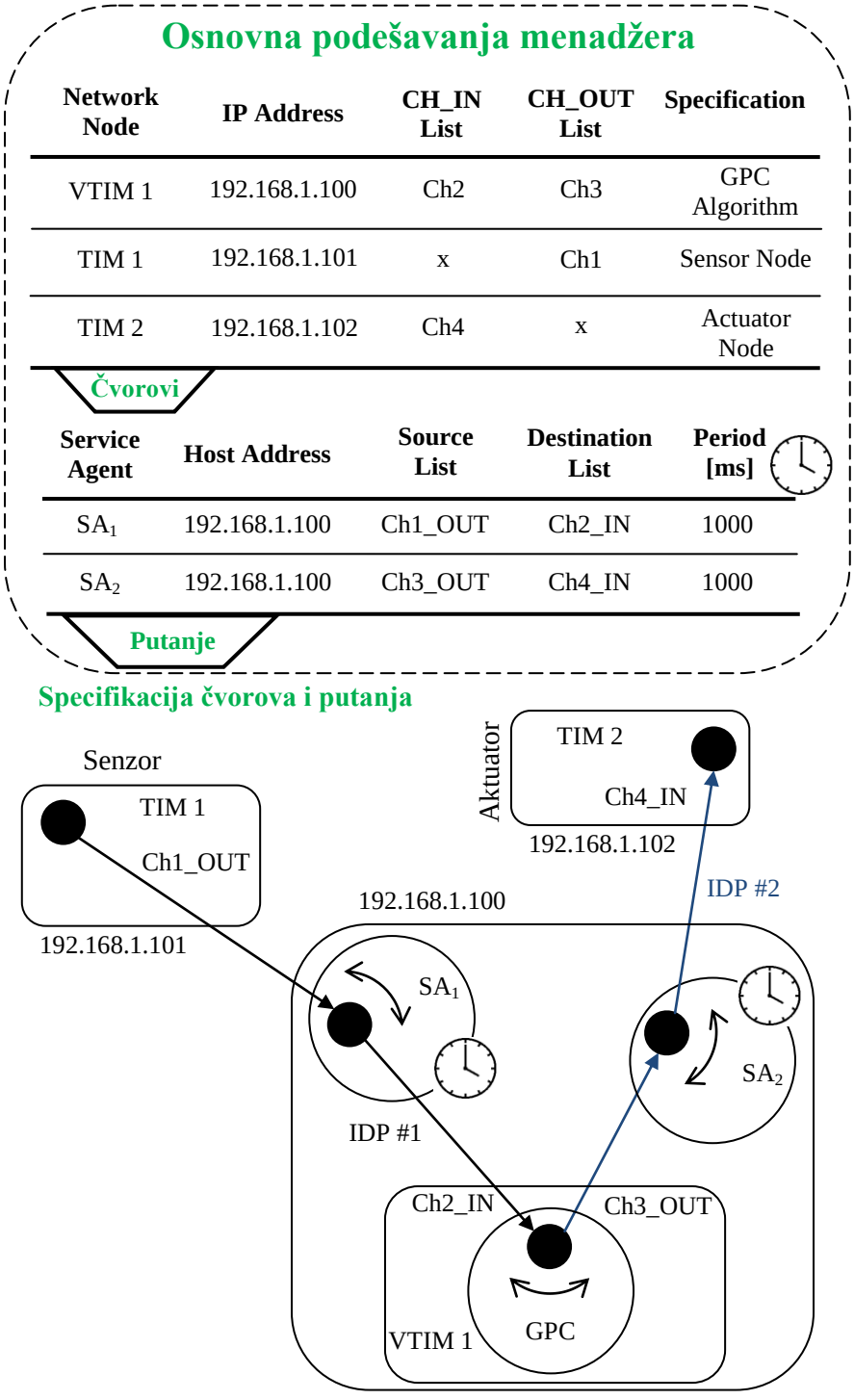
Po prijemu odgovora servisa za čitanje podataka, SA poziva servis za upis podataka na *DST* strani koja je definisana SAEDS-om. Poruka odgovora servisa za upis podataka potvrđuje uspešnost IDP transfera. Ukoliko je omogućena kontrola toka putem parametra *Flow_Control* i odgovor servisa za upis potvrdi uspešnost transfera podataka, *Reference_Time_Stamp* se postavlja na vrednost *Transfer_ID* parametra uključenog u dati odgovor servisa za upis podataka. Na ovaj način, uzastopni pozivi servisa za čitanje, uzrokovani događajima tajmera, zahtevaće samo podatke dostupne posle vremenskog trenutka definisanog *Reference_Time_Stamp* parametrom. Ukoliko je *Flow_Control* vrednost postavljena tako da onemogući kontrolu toka, parametar *Reference_Time_Stamp* se ažurira odmah po prijemu odgovora servisa čitanja, bez obzira na status servisa za upis. Bez obzira na to što se pri pozivu servisa upisa i čitanja na strani SA komponente podrazumeva enkapsulacija vremenskog prozora zahtevanih i isporučenih podataka, respektivno, na implementaciji datih servisa je da obezbedi informacije koje odgovaraju takvom vremenskom prozoru.

Zbog prisutnosti nedeterminističkog kašnjenja u mreži, koje se očekuje prilikom prijema odgovora servisa, *Transfer_ID* se dodatno koristi kako bi se izbegao suvišni IDP transport, odnosno, svi odgovori servisa čitanja se odbacuju od strane SA, ako se odnose na zahtev generisan pre zahteva označenog trenutnom vrednošću *Transfer_ID* parametra.

U prvom koraku, menadžer prikuplja TEDS podatke svih registrovanih uređaja preko CDP putanje prikazane na slici 4.7 i predstavlja te podatke

operateru perko GUI komponente. Radi jasnoće prikaza, slika 4.9 ilustruje specifikaciju čvorova i putanja na strani menadžera, kao i ogoljenu verziju dijagrama sa slike 4.7 koji prikazuje IDP putanje. Operater pristupa osnovnim podešavanjima menadžera i stiče uvid u dostupnost određenih senzora, aktuatora i algoritama, pre nego što definiše putanje podataka koje ih povezuju. U formi tabele su prikazana tri mrežna čvora i njihove IP adrese u okviru *IP Address* polja, liste ulaznih i izlaznih kanala, kao i osnovni opisi. Putanja podataka je definisana pomoću SAEDS-a, koji između ostalog obuhvata listu ulazni/izlaznih kanala i periodu za svaki SA, i ova konfiguracija se preko CDP putanje učitava u mrežni čvor sa adresom navedenom u *Host Address* polju. Sa obzirom na to da studija slučaja podrazumeva upotrebu GPC algoritma, dalji opis operacija će biti dat u skladu sa tom činjenicom.

Kada menadžer pošalje signal inicijalizacije GPC VTIM čvoru preko CDP putanje, startuje se direktna komunikacija između mrežnih čvorova posredstvom SA komponenti. Generalno, svaka SA komponenta može vršiti više od jednog read/write poziva lokalnog/udaljenog servisa sa predefinisanim periodom. Broj SA komponenti zavisi od broja mrežnih čvorova uključenih u IDP putanju i načina na koji operater želi da rasporedi pozive servisa (npr. očitavanje podataka sa dva senzorska čvora može da bude poverenom jednom SA ili se može realizovati pomoću zasebnih SA komponenata). U prikazanom slučaju podaci se periodično čitaju iz kanala senzorskog čvora *Ch1_OUT* preko *IDP #1*, i prenose ka kanalu GPC čvora *Ch2_IN*. Prilikom svakog transfera, SA_1 vrši jedno mrežno čitanje preko STWS klijentskog interfejsa i jedan upis preko lokalnog IEEE 1451.0 sloja. SA_2 periodično očitava rezultat izvršavanja GPC algoritma (kada je dostupan) iz kanala *Ch3_OUT*, a zatim ga isporučuje kanalu aktuatorskog čvora *Ch4_IN* preko putanje *IDP #2*. Prilikom svakog transfera, aktivna komponenta SA_2 vrši jedno čitanje preko lokalnog IEEE 1451.0 interfejsa i jedan mrežni upis putem STWS klijentskog interfejsa.



Slika 4.9 Osnovna podešavanja menadžera i putanje razmene podataka.

Svaki mrežni čvor na putanji podataka obezbeđuje vremensku referencu u formi brojača sa rezolucijom 1ms, koja može biti upotrebljena za sledeće operacije: za pridruživanje vremenske reference senzorskim podacima, za

podudaranje lokalnog brojača i vremenske reference odbiraka u paketu podataka primljenom na strani aktuatora, kao i za *time-aware* procesiranje na strani GPC VTIM mrežnog čvora.

Prikazana tri čvora zatvaraju komunikacioni lanac preko *IDP #1* i *IDP #2* putanja, što obezbeđuje optimalni mrežni transport bez posredničkog čvora. Servis za vremensku sinhronizaciju je uveden u STWS sloj da obezbedi sinhronizaciju brojača svakog od uređaja i bazira se na slanju sinhronizacionih poruka od strane menadžera perko CDP putanje, upotrebom *multicast* mehanizma. Model seta podataka je uniforman za sve mrežne čvorove i podrazumeva uključivanje vremenske reference u IDP pakete podataka, dok je format seta podataka za svaki kanal mrežnog čvora definisan TEDS sadržajem. Paketi podataka koji sadrže vremensku referencu prenose se duž IDP putanje, tako da čvorovi na putanji vrše *time-aware* procesiranje. Iako to nije predmet dalje analize, potrebno je istaći da uniforman model seta podataka i postojanje vremenskih brojača daju osnovu za realizaciju *multi-rate* operacija mrežnih čvorova.

B. Implementacija i interakcija sistemskih komponenata

I pored toga što svaka pasivna komponenta distribuiranog sistema mora da obezbedi podršku za servisno orijentisanu komunikaciju na putanjama podataka, akcije koje se tiču samog prenosa podataka su enkapsulirane unutar SA aktivnih komponenata. Interakcija SA i ostalih komponenata je detaljnije analizirana u nastavku, jer predstavlja ključ realizacije usvojenog modela transfera podataka.

Senzorski čvor se implementira prema IEEE 1451.0 konceptu. Pri tome, informacija o vremenskoj referenci koja nije posebno razmatrana IEEE 1451 standardom, je pridružena odbirku koji se prenosi u okviru seta podataka. Kao deo mehanizma vremenske sinhronizacije na sistemskom nivou, senzorski čvor reguliše lokalni vremenski brojač i obezbeđuje podršku za njegovu konfiguraciju od strane menadžerskog čvora preko posebnog vremenskog servisa.

Aktuatorski čvor, slično kao i senzorski, podržava IEEE 1451 koncept i održavanje lokalnog brojača. Sa druge strane, ulazni podaci aktuatora su dati u formi paketa podataka koji sadrži set kontrolnih inkremenata sa vremenskom referencom. U tom slučaju se javlja potreba za baferisanjem kontrolnih inkremenata u formatu koji uključuje i vremenski trenutak pridružen svakom inkrementu. Kontrolna logika takvog bafera, omogućava ispisivanje odbiraka prema vremenskoj poziciji u baferu i to samo za pakete sa najnovijim *timestamp* vrednostima. Operacija čitanja odgovarajućih inkremenata iz bafera se odvija u skladu sa lokalnim vremenom aktuatora.

Kao što je već rečeno, GPC čvor obuhvata implementaciju GPC algoritma u formi VTIM-a, kao i servisne agente. U nastavku će biti dat opis implementacionog modela GPC algoritma sa baferisanim ulazom/izlazom. Nakon konfiguracije, TEDS GPC čvora sadrži parametre horizonta predikcije n_y, horizonta kontrole n_u, težinskog koeficijenta kontrolnih inkremenata λ, kao i koeficijente a_i i b_i diskretnog modela procesa:

$$G(z^{-1}) = \frac{\sum_{j=1}^{n} b_j z^{-j}}{1 + \sum_{i=1}^{n} a_i z^{-i}} \tag{4.1}.$$

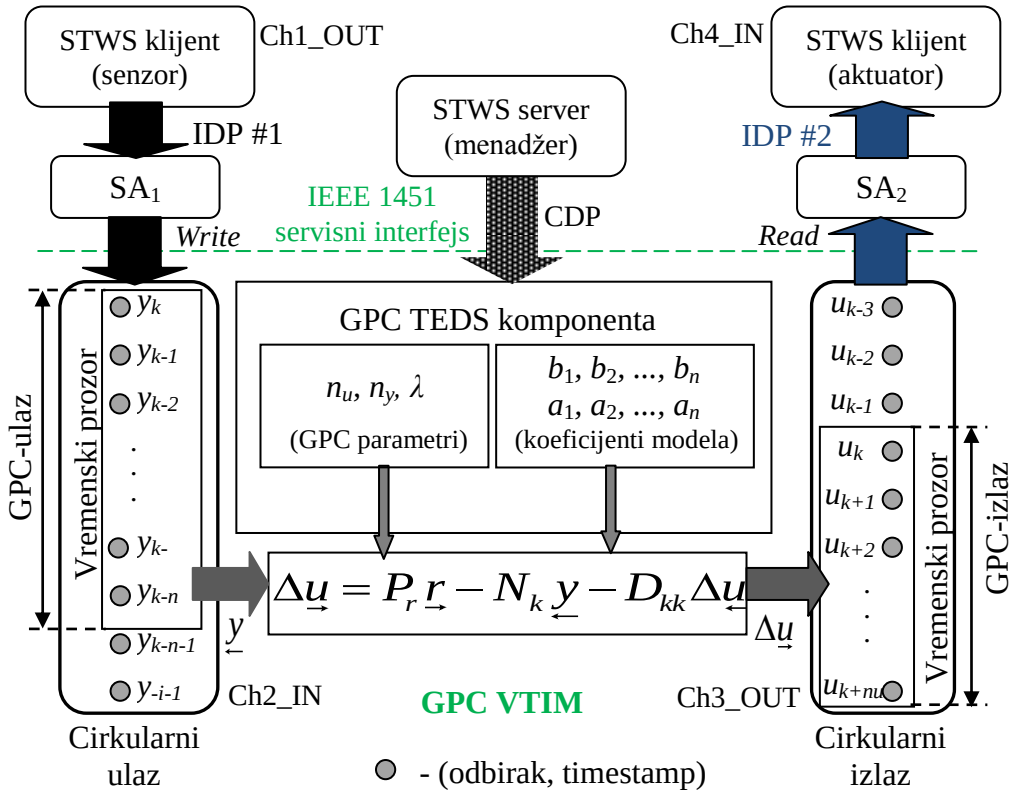
Kako bi se kompenzovalo kašnjenje na putanji podataka, usvaja se GPC kontrolni algoritam baziran na opisu modela procesa u formi transfer funkcije [45], u modifikovanom obliku:

$$\Delta \underset{\rightarrow}{u} = P_r \underset{\rightarrow}{r} - N_k \underset{\leftarrow}{y} - D_{kk} \Delta \underset{\leftarrow}{u} \tag{4.2},$$

kako bi se dobio vektor $\Delta \underset{\rightarrow}{u} = [\Delta u_k, \Delta u_{k+1}, \ldots \Delta u_{k+n_u}]$ od ukupno n_u budućih kontrolnih inkremenata, gde se koristi notacija sa strelicama ispod simbola da se opišu prikupljeni merni odbirci $\underset{\leftarrow}{y}$, prethodno sračunati kontrolni inkrementi $\Delta \underset{\leftarrow}{u}$ i buduće vrednosti reference $\underset{\rightarrow}{r}$, respektivno: $\underset{\leftarrow}{y} = [y_k, y_{k-1}, \ldots y_{k-n}]$, $\Delta \underset{\leftarrow}{u} = [\Delta u_{k-1}, \Delta u_{k-2}, \ldots \Delta u_{k-n}]$, i $\underset{\rightarrow}{r} = [r_{k+1}, r_{k+2}, \ldots r_{k+n_y}]$. Matrice P_r, N_k, i D_{kk} su iste u

svakoj iteraciji i sračunavaju se jednom, pre prvog izvršavanja GPC algoritma na osnovu parametara algoritma i modela procesa.

U vremenu regularnog rada GPC VTIM prihvata pakete sa podacima i vremenskom referencom preko *IDP #1* i smešta odbirke u cirkularni ulazni bafer kao na slici 4.10. Kontrolni algoritam uzima odgovarajući vremenski prozor $\underleftarrow{y}$ senzorskih merenja i upisuje $\underrightarrow{\Delta u}$ kao rezultat algoritma u izlazni cirkularni bafer. Rezultujući odbirci se šalju pametnom aktuatoru preko *IDP #2*, koji prema timestamp podatku bira najpogodniji inkrement i sračunava vrednost za apliciranje na izlazu.



Slika 4.10 GPC algoritam sa procesiranjem odbiraka dobijenih iz paketa podataka sa vremenskom referencom.

GPC koji se implementira kao zasebna nit, prihvata ulazne podatke na bazi događaja lokalnog servisnog zahteva, a kasnije upisuje rezultat algoritma u izlazni kanal *Ch3_OUT* pri čemu poziva *notify()* funkciju za aktivaciju SA_2, što

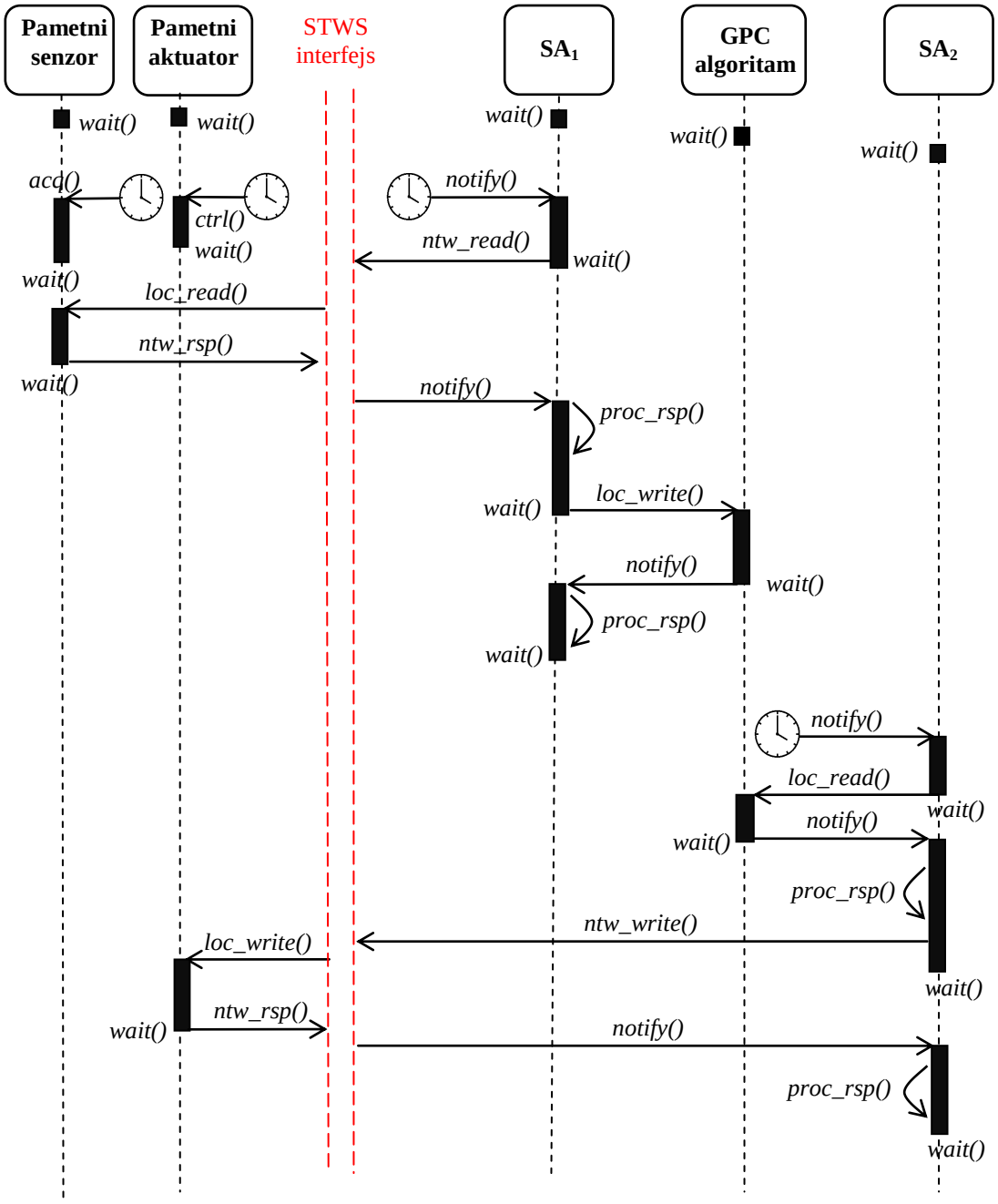
će kasnije biti detaljnije objašnjeno. Vremenski prozor na izlazu mora biti dovoljno dugačak da kompenzuje kumulativno kašnjenje putanja *IDP #1* i *IDP #2*. GPC VTIM podrazumeva da nema nedostajućih podataka u ulaznom baferu, što zahteva da bude omogućena kontrola toka komponente SA_1, setovanjem SAEDS parametra *Flow_Control* (slika 4.8).

Naknon konfiguracije svih komponenata u mreži od strane menadžera, sinhronizuju se vremenski brojači čvorova i startuje se regularna razmena podataka aktiviranjem SA komponenata. Dijagram sekvenci koji prikazuje operacije svih komponenata na nivou programskih niti u trajanju od jedne periode regularnog rada sistema, prikazan je slikom 4.11. U prikazanom slučaju oba servisna agenta SA_1 i SA_2 se izvršavaju na GPC VTIM čvoru, tako da postoji razlika između udaljenih *read/write* poziva servisa prema pametnom senzoru/aktuatoru preko STWS interfejsa, označenih sa *ntw_read()*/*ntw_write()*, i lokalnih poziva GPC servisa, označenih sa *loc_read()*/*loc_write()*. Najčešći je slučaj da nit čeka na notifikaciju povodom nekog događaja, čime se inicira akcija niti u skladu sa tim događajem.

Lokalni tajmeri notifikuju odgovarajuće niti nodova kako bi se iniciralo procesiranje događaja tajmera na početku periode. Za pametne pretvarače to podrazumeva obavljanje poslova akvizicije i kontrole što je na dijagramu označeno pozivima funkcija *acq()* i *ctrl()*, respektivno. Notifikacije tajmera u slučaju niti SA_1 i SA_2 podrazumevaju startovanje transfera podataka duž IDP putanja, kroz slanje zahteva *read* servisa STWS/lokalnog interfejsa.

Slanje zahteva pozivom *ntw_read()* funkcije od strane niti SA_1, se preko STWS interfejsa efektivno preslikava na poziv *loc_read()* funkcije na strani pametnog senzora. Odgovor senzora se vraća pozivom funkcije *ntw_resp()*, koja dovodi do notifikacije SA_1 niti kroz STWS interfejs. Nakon toga, podaci se prenose ka GPC niti pozivom *loc_write()* čime se startuje izvršavanje algoritma. Nakon prijema podataka i njihovog smeštanja u ulazni bafer, GPC nit šalje i potvrdu prijema, čime se omogućava SA_1 niti da ažurira ranije opisane parametre kontrole toka (*Transfer_ID* i *Reference_Time_Stamp*). Na strani niti SA_2

se koristi sličan mehanizam za čitanje rezultata izvršavanja GPC algoritma i slanje inkremenata upravljanja pametnom aktuatoru.



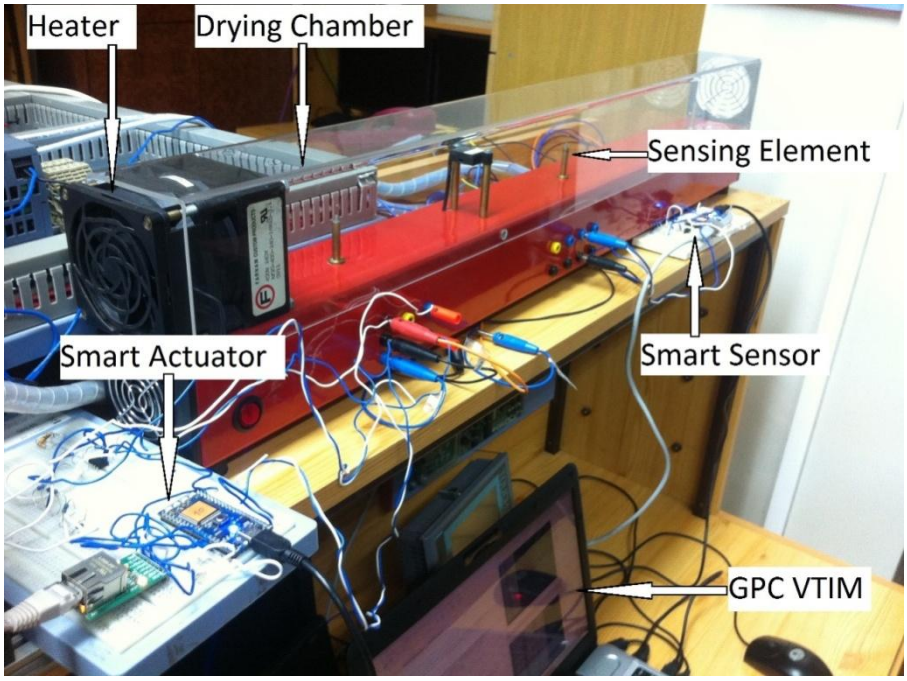
Slika 4.11 Dijagram izvršavanja operacija različitih sistemskih komponenata na nivou programskih niti, u toku jedne periode regularnog rada.

Prikazano ponašanje podrazumeva regularnu sekvencu događaja i samim tim regularan redosled izvršavanja poslova pojedinačnih niti. Međutim,

slučajno mrežno kašnjenje i gubici paketa na IDP putanjama mogu dovesti do promene redosleda analiziranih događaja i izostanka paketa odgovora servisa, respektivno.

Sa obzirom na to da je funkcionalnost prenosa podataka sadržana u funkcijama servisnih agenata, to su komponente koje su podložne mrežno-indukovanim problemima. Akcije SA komponenata u tim situacijama su već definisane prikazanim modelom izvršavanja operacija. I pored toga, u nastavku je data analiza performansi upravljačke petlje u prisustvu mrežno-indukovanih problema.

Dati model je implementiran u slučaju kontrole temperature sušare. Laboratorijska postavka prikazana slikom 4.12 obuhvata sušaru (proces) i sve relevantne mrežne čvorove kontrolnog sistema. Kontrolna promenljiva procesa je kontrolni napon grejača u opsegu od [0V, 10V], koji se kondicionira i dovodi na grejač pomoću posredničkog kola za pojačanje snage.



Slika 4.12 Laboratorijska postavka za testiranje modela sa upotrebom servisnih agenata.

Promenljiva koja se reguliše je temperatura sušare, praćena Pt100 senzorskim elementom, tako da se naponski ekvivalent temperature posebnim kolom za kondicioniranje takođe svodi na opseg napona [0V, 10V]. Analizom karakteristika sušare, dolazi se do preliminarne identifikacije modela prvog reda i dinamičkih karakteristika procesa:

$$G(s) = \frac{K}{T_p s + 1} e^{-\tau s},\ K \in [0.05, 0.4],\ T_p \approx 4\,\text{sec},\ \tau \approx 1\,\text{sec}, \tag{4.3}$$

gde je dinamika (vremenska konstanta T_p i mrtvo vreme τ), praktično nezavisna od uslova rada, a dominantna nelinearnost se ogleda u promeni pojačanja K u zavisnosti od temperature sušare. Pojačanje K uzima visoke vrednosti pri srednjem opsegu temperature (oko 45^0C), ali pada na niske vrednosti na krajevima posmatranog opsega temperature: na ambijentalnoj temperaturi od približno 25^0C i visokim vrednostima temperature do 60^0C. Takođe, posledice procesa hlađenja i grejanja nisu iste za pojačanje K sa obzirom na postojanje histerezisa. Kako bi se dobio model procesa prihvatljiv za širok opseg temperaturne kontrole, usvajaju se perioda odabiranja i parametri modela (4.3) na sledeći način:

$$T_s = 1\ \text{sec},\ K = 0.2,\ T_p = 4\ \text{sec},\ \tau = 1\ \text{sec} \tag{4.4}$$

tako da se za potrebe prediktivnog upravljanja konačno dobija diskretni model sa zadržavanjem nultog reda:

$$G(z^{-1}) = \frac{b_2 z^{-2}}{1 + a_1 z^{-1}},\quad a_1 = -e^{-T_s/T_p} = -0.7788,\quad b_2 = K\left(1 - e^{-T_s/T_p}\right) = 0.0442 \tag{4.5}.$$

Parametri GPC kontrolnog algoritma su izabrani na sledeći način

$$n_u = 10,\ n_y = 10,\ \lambda = 0.3 \tag{4.6}$$

gde usvojeni horizont budućih kontrolnih odbiraka n_u obezbeđuje pouzdan rad sistema sa mrežnim kašnjenjem koje ne prelazi 10 perioda odabiranja, dok je horizont predikcije n_y biran tako da pokrije vreme uspona u otvorenoj petlji.

Vrednost težinskog koeficijenta λ je podešena tako da ograniči amplitude inkremenata upravljanja na jediničnu vrednost.

Pametni senzor i aktuator su razvijeni na bazi ARM prototipa generičkog modula opisanog u sekciji 3.1, dok je GPC čvor implementiran na platformi opšte namene u Javi, korišćenjem razvojnog okruženja NetBeans IDE. Pritom, isti set STWS servisa se izvršava na svakom od namenskih čvorova i na GPC čvoru. Menadžerski čvor je implementiran u vidu posebnog procesa sa HTML GUI interfejsom i Java DB bazom podataka za čuvanje TEDS sadržaja. U prikazanoj eksperimentalnoj postavci, pametni pretvarači i GPC čvor razmenjuju SOAP/XML poruke u mreži preko jednostavnog LAN rutera namenjenog kućnoj upotrebi.

Slike 4.13 i 4.14 prikazuju rezultate testiranja mreže u slučaju kontrole temperature sušare. Scenario testiranja je definisan tako da uključuje efekte gubitka paketa i nepredvidljivog kašnjenja koji se mogu javiti u realnim situacijama. Mrežno kašnjenje je simulirano u opisanoj postavci zadržavanjem paketa na strani GPC čvora tokom slučajno izabranog vremenskog intervala. Gubitak paketa je simuliran sukcesivnim odbacivanjem određenog broja paketa počevši od nekog predefinisanog vremenskog trenutka. Podrazumevani period izvršavanja SA komponente je isti kao i perioda odabiranja T_s koja se koristi za kontrolu i akviziciju. Sa obzirom na to da se svi čvorovi nalaze u lokalnoj mreži, uticaj kašnjenja i gubitka paketa na rad sistema u realnom vremenu nije značajan, zbog čega se i pribegava uvođenju veštačkog kašnjenja i gubitka paketa. Dati rezultati pokazuju uticaj mrežno-indukovanih problema na rad sistema u realnom vremenu.

U prvom scenariju, slučajno kašnjenje se ubacuje u informacionu putanju podataka IDP #2, od strane servisnog agenta SA_2, prema arhitekturi prikazanoj na slikama 4.7 i 4.9. Na taj način se dobija kašnjenje paketa kontrolnih inkremenata koje generiše GPC čvor, a koje prima i dalje procesira aktuator. Na osnovu lokalnog vremenskog brojača i vremenske reference u okviru paketa, aktuator iz pristiglog paketa bira odgovarajuće inkremente upravljanja i aplicira

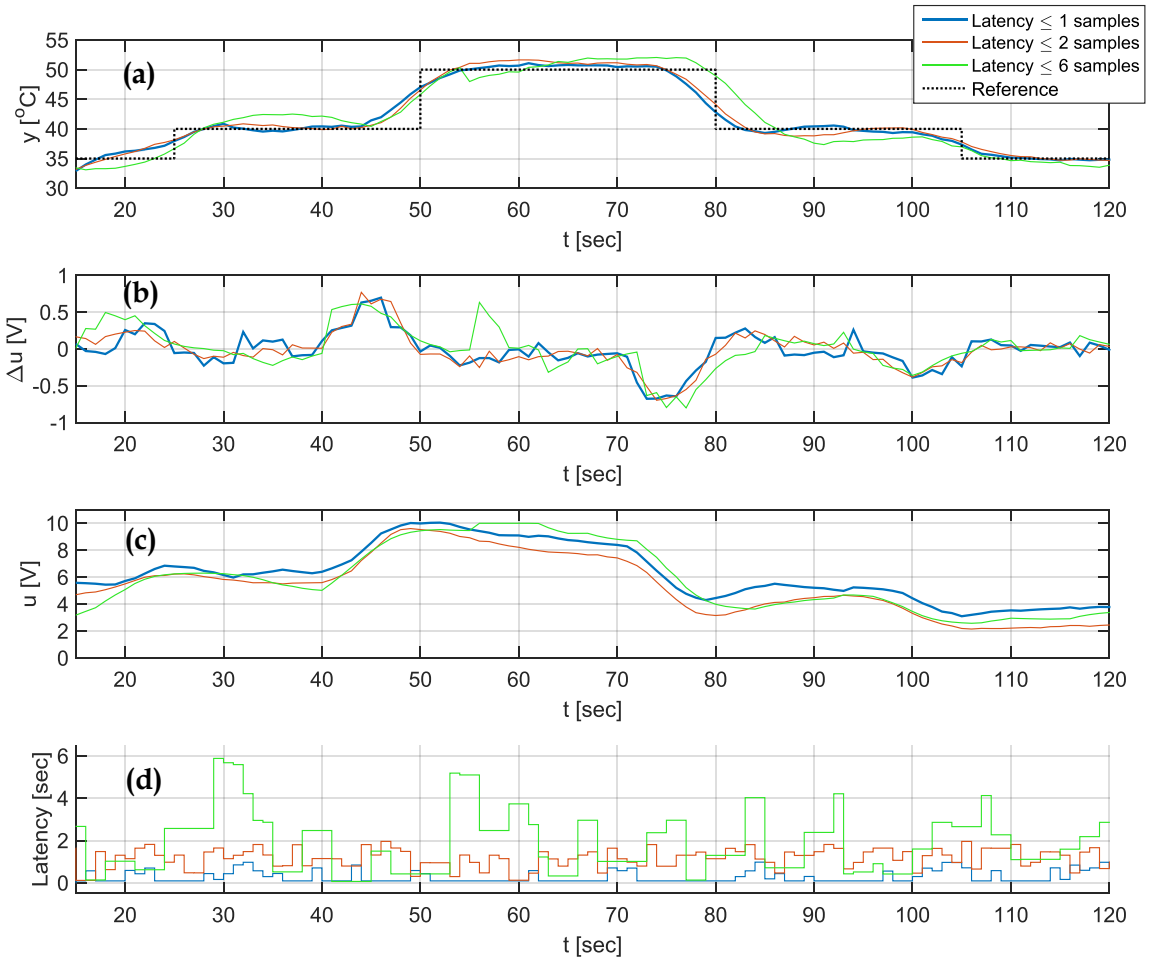
konačni odbirak kontrolnog signala. Veće kašnjenje na putanji IDP #2 rezultuje biranjem vremenski daljih odbiraka iz prediktivnog seta inkremenata, pa samim tim i drugačijom vrednošću konačnog kontrolnog odbirka. Pritom, isti efekat se očekuje i prilikom ubacivanja slučajnog kašnjenja na putanji IDP #1, jer se i u tom slučaju dobija isto ukupno kašnjenje kao zbir kašnjenja po putanjama IDP #1 i IDP #2.

Slike 4.13(c) i 4.13(b) prikazuju kontrolni signal i inkremente upravljanja na strani aktuatora, respektivno, za sledeća tri scenarija simulacije mrežnog kašnjenja (označenog na slici sa *Latency*):

- Bez slučajnog kašnjenja. IDP kašnjenje paketa (kašnjenje duž putanja IDP #1 i IDP #2) je dato bez ubacivanja dodatnog kašnjenja od strane SA_2.
- Slučajno kašnjenje je manje od 1s. U ovom slučaju, kašnjenje koje predstavlja celobrojni umnožak od 100ms se ubacuje u IDP putem GPC čvora. Dodatno kašnjenje je manje od 1s.
- Slučajno kašnjenje je manje od 10s. U ovom slučaju, kašnjenje koje predstavlja celobrojni umnožak od 100ms se ubacuje u IDP putem GPC čvora, isto kao u prethodnom slučaju. Dodatno kašnjenje je manje od 10s.

Slika 4.13 (a) prikazuje signal senzora, što je temperatura u stepenima Celzijusa podeljena sa 10, kao i zadati referentni signal, dok slika 4.13 (d) prikazuje trenutno kašnjenje za opisana tri scenarija. Treba primetiti da mrežno kašnjenje prikazano na dijagramu ne prelazi 6s, iako slučajno ubačeno kašnjenje dostiže vrednost od čak 10s. Ovo je postignuto implementacijom mehanizma filtriranja na strani aktuatora. Pametni aktuator uzima uvek najnoviji paket, pri čemu odbacuje sve ostale pakete, što se postiže analizom vremenske reference u formi ranije opisanog *Transfer_ID* parametra. U tom slučaju većina paketa sa značajnim mrežnim kašnjenjem se odbacuje, što je pokazano u toku testiranja. Senzorski i aktuatorski podaci, kao i IDP kašnjenje su snimljeni posebno za svaki od scenarija slučajnog kašnjenja na intervalu od 110s, sa sledećim nivoima

referentnog signala za setovanje željene temperature sušare: 35-40-50-40-35°C.



Slika 4.13 Simulacija slučajnog mrežnog kašnjenja u slučaju kontrole temperature sušare: (a) Dobijena temperatura i zadata referenca; (b) Inkrementi upravljanja na strani aktuatora; (c) Kontrolni signal primenjen od strane aktuatora; (d) Slučajno mrežno kašnjenje ubačeno od strane SA_2.

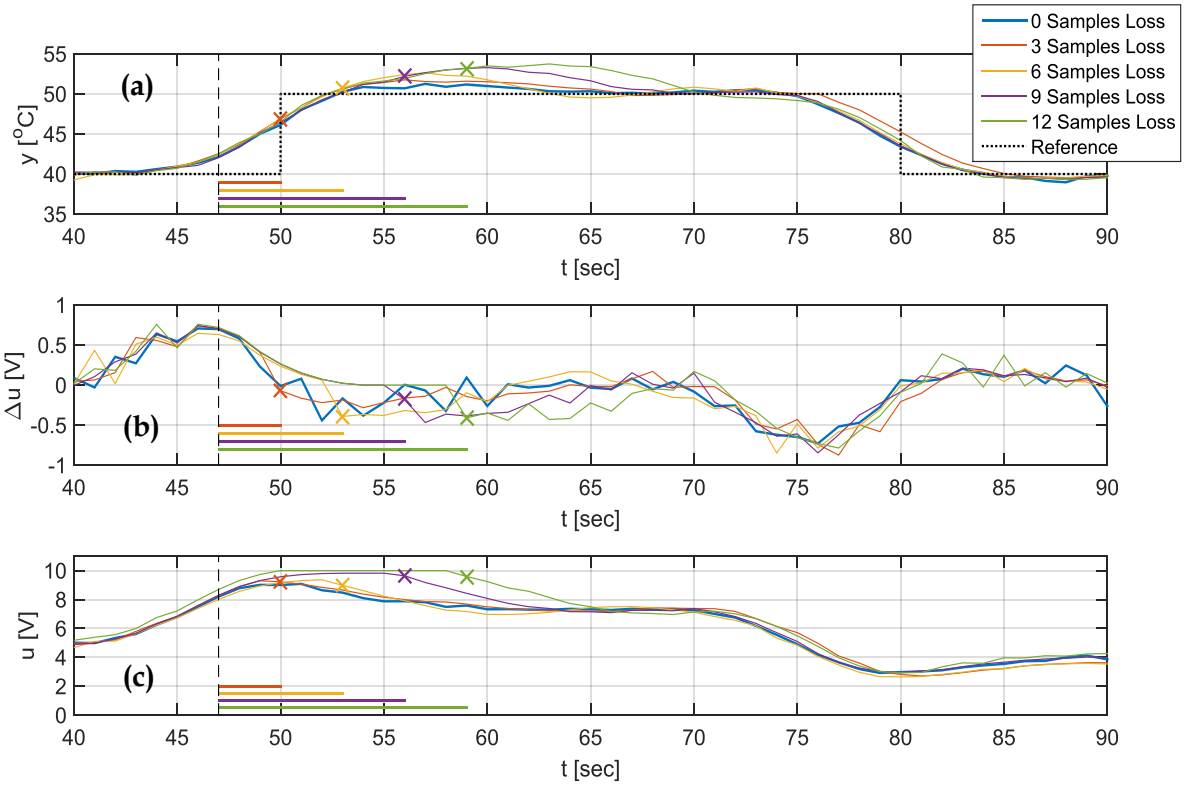
Dijagrami prikazuju rezultate rada u realnom vremenu sa periodom odabiranja od 1s, dajući trenutne vrednosti senzorskih/aktuatorskih podataka i transportnog kašnjenja. Slika 4.13(b)-(c) prikazuje uticaj mrežnog kašnjenja na trenutno primenjeni kontrolni inkrement Δu_k i kontrolni signal u dobijen akumulacijom datih inkremenata. U svakom kontrolnom koraku, odnosno u

svakoj sekundi, jedan kontrolni inkrement iz novodobijenog paketa se dodaje na postojeću vrednost u. Ukoliko novi paket izostane, aktuator primenjuje sledeći kontrolni inkrement iz postojećeg paketa i tako ažurira vrednost u. IDP transportno kašnjenje se procenjuje na strani aktuatora kao razlika između lokalnog vremenskog brojača i vremenske reference koja se pridružuje mernom odbirku od strane senzora. Transportno kašnjenje paketa, prikazano slikom 4.13(d) se smatra konstantnim tokom akumuliranja kontrolnih inkremenata Δu_k u okviru istog paketa, sve do pristizanja novog paketa i izračunavanja nove vrednosti transportnog kašnjenja. Zadržavanja paketa u mreži u toku određenog vremena, primorava aktuatorski čvor da aplicira GPC inkremente iz prethodno primljenog paketa.

U scenariju sa malim mrežnim kašnjenjem (manjim od 1s), na slici 4.13(a) se vidi da senzorski signal ne odstupa mnogo od referentnog signala, dok ostali scenariji pokazuju nešto značajnije odstupanje. U trećem slučaju gde IDP vrednost slučajnog kašnjenja dostiže 6s, temperatura odstupa više od zadate reference nego u prethodnim slučajevima, kao što je i očekivano, jer greška GPC inkremenata raste kako aktuator iterira kroz primljeni set podataka u odsustvu paketa koji kasni. Pokazano je da sistem zadržava stabilnost, čak i u slučaju velikog IDP kašnjenja. Pritom, mrežno kašnjenje je kompenzovano predikcijom budućih kontrolnih inkremenata Δu_k upotrebom GPC algoritma, kao i sinhronizacijom na bazi vremenske reference modela podataka. Data tehnika kompenzacije kašnjenja omogućava implementaciju sistema koji zadržava i kvalitet SOA interoperabilnosti i karakteristike sistema za rad u realnom vremenu koje su potrebne u merno-kontrolnim aplikacijama.

Slika 4.14 prikazuje signal senzora, kontrolne inkremente i kontrolni signal na strani aktuatora za šest scenarija gubitka paketa, slično kao u slučaju scenarija slučajnog mrežnog kašnjenja. Dijagram 4.14(a) prikazuje signal senzora i vremenske trenutke početka i završetka sekvence gubitka paketa za svaki od scenarija. Kada dođe do gubitka paketa, aktuator primenjuje kontrolne inkremente iz postojećeg seta podataka, slično kao u slučaju kašnjenja paketa.

Simulacija gubitka paketa počinje 3s pre promene vrednosti referentne temperature sa 40^0C na 50^0C. Na taj način, simulacija prekida regularan rad sistema i uvodi period bez komunikacije. Senzorski i aktuatorski podaci su snimljeni posebno za svaki scenario gubitka paketa, na intervalu od 90s, sa sledećim zadatim nivoima referentne temperature sušare: 40-50-40-35°C.



Slika 4.14 Simulacija gubitaka paketa u slučaju kontrole temperature sušare: (a) Dobijena temperatura i zadata referenca; (b) Inkrementi upravljanja na strani aktuatora; (c) Kontrolni signal primenjen od strane aktuatora.

Horizontalne linije u donjem delu svakog od prikazanih dijagrama, označavaju početak i kraj sekvence bez komunikacije, pri čemu je svaki scenario označen posebnom bojom. Na datom vremenskom intervalu, aktuator sukcesivno primenjuje postojeće kontrolne inkremente primljene pre trenutka gubitka komunikacije. Gubitak komunikacije se manifestuje odstupanjem senzorskog signala u odnosu na zadatu referencu, kao što se vidi na slici 4.14(a). U scenarijima gde gubitak komunikacije traje više od 10s, aktuator ostaje bez

inkremenata (jer je n_u postavljeno na 10), pa nastavlja da ažurira vrednost u dodajući na tu vrednost poslednji dati inkrement u svakom sledećem koraku upravljanja. Ova situacija je prisutna kada novi paket izostaje na intervalu dužine 12s, što dovodi do većeg odstupanja tokom tranzicije temperature u odnosu na druge slučajeve.

Znak 'x' na dijagramima označava kraj sekvence bez komunikacije, odnosno trenutak ponovnog uspostavljanja kontrole. Nakon uspostavljanja komunikacije distribuirana kontrola stabilizuje senzorski signal za svaki od prikazanih scenarija. Pokazano je da se gubitak paketa kompenzuje pomoću GPC algoritma i sinhronizacije na bazi modela podataka sa vremenskom referencom, slično kao u scenariju sa slučajnim kašnjenjem u mreži. Zahvaljujući tome, omogućena je realizacija robustnih sistema baziranih na SOA arhitekturi, sposobnih da zadrže kontrolu tokom kratkih vremenskih perioda sa izostankom komunikacije.

U ovoj sekciji, prikazana je distribuirana servisno orijentisana arhitektura za upravljanje u mreži pametnih pretvarača i data je studija slučaja kontrole temperature sušare. Dati model omogućava, dislokaciju funkcionalnosti među mrežnim čvorovima i automatizaciju razmene podataka upotrebom servisnih agenata, čija mogućnost dislokacije obezbeđuje realizaciju proizvoljne mrežne topologije. Konfiguracija mreže koju pruža predloženi SAEDS model, omogućava efikasnu upotrebu servisnih agenata kao aktivnih komponenata koje obavljaju periodičnu TIM-VTIM komunikaciju i raščlanjavaju ulazno-izlazne poslove i izvršavanje algoritama, smeštajući ih u zasebne niti izvršavanja. Servisni agenti se koriste kao vezivne komponente koje se oslanjaju na jednostavan STWS klijentski interfejs, pa su samim tim pogodni i za implementaciju na namenskim uređajima sa ograničenim memorijskim resursima. Vremenska sinhronizacija je obezbeđena upotrebom vremenskih referenci zajedno sa senzorskim i aktuatorskim podacima, upotrebom GPC algoritma, mehanizmom filtriranja na strani pametnog aktuatora i dodatnim STWS servisom za sinhronizaciju. Mrežni problemi varijabilnog kašnjenja i

gubitka komunikacije su kompenzovani predikcijom kontrolnih inkremenata putem GPC algoritma. Dato rešenje obezbeđuje podršku za implementaciju distribuiranih merno-upravljačkih sistema sa karakteristikama sistema za rad u realnom vremenu, uz upotrebu gotovih hardverskih i softverskih komponenti.

5. POTENCIJAL PRIMENE PREDLOŽENOG MODELA

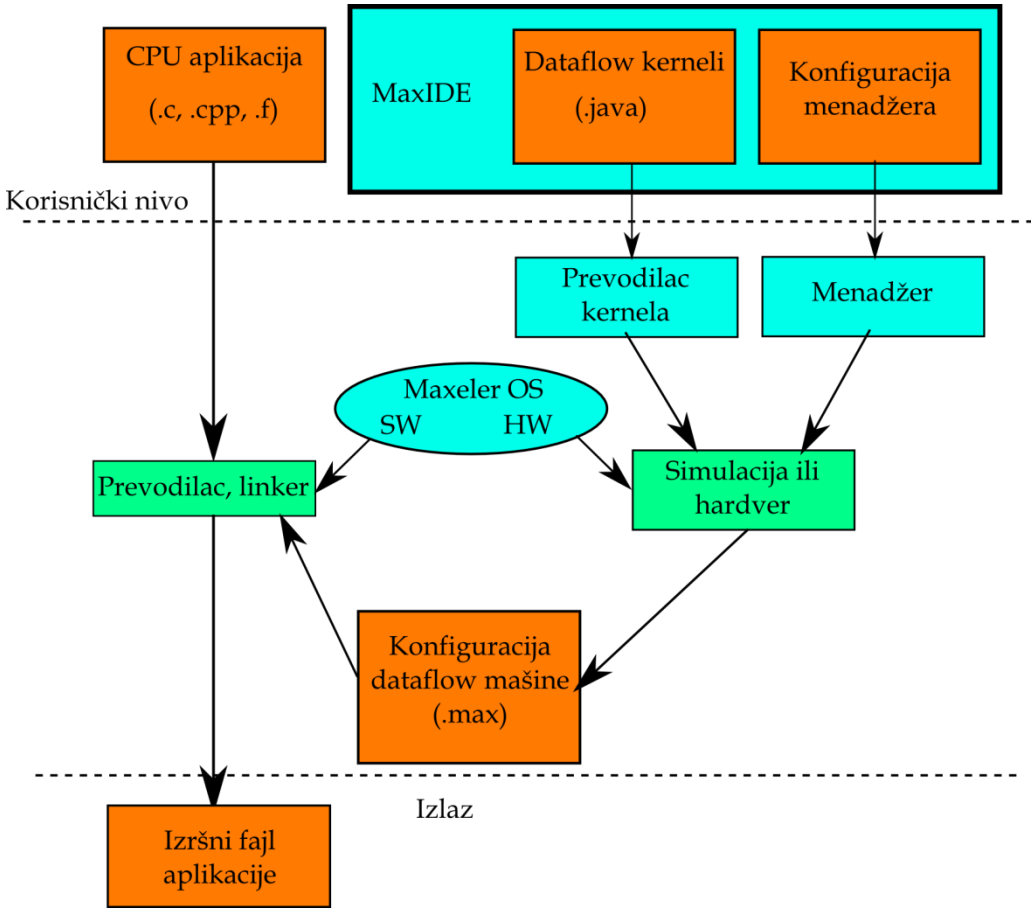
Upotreba predložene arhitekture nije ograničena na distribuirane kontrolne sisteme u kojima je komunikacija bazirana na SOAP/XML protokolu i HTTP transportnom sloju. VTIM komponenta ne daje praktično nikakva ograničenja po pitanju vrste mrežnog entiteta koji se integriše u mrežu, dok aktivne komponente u vidu servisnih agenata nemaju restrikcije po pitanju upotrebe transportnog komunikacionog protokola. U nastavku će biti data ilustracija koncepta integracije akceleratora RSA enkripcije u VTIM formi, kao i primer upotrebe servisnog agenta na bazi SNMP (Simple Network Management Protocol) mrežnog protokola. Takođe, biće data i diskusija o različitim aspektima implementacije predloženog servisno orijentisanog modela u opštem slučaju. Na kraju poglavlja, analizirana je mogućnost upotrebe datog modela prema principima *Cloud Computing*-a i *Internet of Things*-a (IoT-a).

5.1. VTIM kao akcelerator enkripcije: RSA algoritam na Maxeler platformi

Sa porastom količine informacija na Internet mreži, sve veći značaj ima deljenje podataka u distribuiranom okruženju iz heterogenih izvora, kao što su senzorski podaci i naučne baze podataka koje mogu razmenjivati istraživačke zajednice i organizacije [46]. Velike količina podataka i njima prilagođenje platforme otvaraju nove perspektive primene algoritama. U okviru ove teze, za potrebe sigurnog deljenja i čuvanja velikih setova podataka razmotrena je upotreba RSA algoritma za enkripciju, koju je teško implementirati na standardnoj CPU arhitekturi bez dobijanja značajnog kašnjenja enkripcije. Kao akcelerator, izabrana je Maxeler platforma [47-50], bazirana na rekonfigurabilnoj FPGA logici za procesiranje nizova podataka prema dataflow arhitekturi sa slike 5.1. Ključni deo platforme predstavljaju *pipeline* strukture sa većim brojem nivoa, za realizaciju komponenata aritmetičkih operacija kao što su sabirači, množači i slično, za procesiranje nizova podataka. Velika dubina *pipeline* struktura zahteva određenu količinu podataka na ulazu kernela za

njihovo adekvatno popunjavanje. Ulazni podaci koji po svojoj veličini ili strukturi ne odgovaraju opisanom konceptu, mogu prouzrokovati i sporije izvršavanje programa u odnosu na slučaj korišćenja standardne CPU arhitekture. Zbog toga je važno pronaći aplikaciju pogodnu za portovanje na ovakvu platformu, pri čemu svaki novi algoritam predstavlja zasebnu studiju slučaja. Opšti je slučaj da aplikacije sa velikom količinom podataka i visokim stepenom paralelizma operacija ostvaruju prednost na Maxeler platformi.

Glavni korisnički program je dat u C/C++ ili Fortran programskom jeziku, pri čemu se delovi čije se ubrzanje zahteva izmeštaju iz glavnog programa u kernele. Kerneli se realizuju u ekstenziji Java jezika za opis hardvera, tako da prevodilac generiše fajl sa detaljima na nivou hardvera pogodan za simulaciju ili za izvršavanje na FPGA čipu.



Slika 5.1 Opšti pregled Maxeler arhitekture.

Menadžer konfiguriše interfejs prema kernelima i njihovo povezivanje sa drugim FPGA čipovima, DRAM memorijom i PCI Express magistralom. Aplikacija host procesora koristi funkcije Maxeler operativnog sistema za konfiguraciju FPGA čipa i transfer podataka. Maxeler prevodilac na taj način vrši mapiranje struktura viših programskih jezika u DFE (DataFlow Engine) strukture na FPGA čipu. Pritom, DFE se konfiguriše pomoću jednog ili više kernela i jednog menadžer fajla. Kerneli definišu hardverske strukture za procesiranje podataka kako bi se postiglo ubrzanje, dok menadžer obezbeđuje komunikaciju kernela i uređaja van čipa.

U datom slučaju, algoritam za enkripciju mora biti prilagođen Maxeler arhitekturi i odgovarajuće opterećenje u vidu ulaznih podataka mora biti obezbeđeno. Prilikom implementacije RSA algoritma, ulazni sadržaj se posmatra kao niz cifara, odnosno kao velika neoznačena celobrojna vrednost. Takav broj zatim ulazi u postupak modularne eksponentizacije kako bi se dobila šifrovana vrednost [51]. Pritom, koristi se Montgomerijev metod redukcije za potrebe poboljšanja performansi modularne aritmetike RSA algoritma, koji se može implementirati na različite načine. Pet načina je predloženo u radu [52], i to su:

- *Separated Operand Scanning (SOS),*
- *Coarsely Integrated Operand Scanning (CIOS),*
- *Finely Integrated Operand Scanning (FIOS),*
- *Finely Integrated Product Scanning (FIPS),*
- *Coarsely Integrated Hybrid Scanning (CIHS).*

U tom slučaju, zaključak autora rada [52] je da CIOS metod jeste najefikasniji ako se razmatra implementacija u C jeziku i asembleru na CPU arhitekturi. Referentna implementacija u okviru ove teze je bazirana na SOS metodu, koji je po svojim karakteristikama najpogodniji za implementaciju na Maxeler platformi i koji se ne razlikuje puno od najefikasnijeg CIOS metoda. Izabrani SOS metod, koji predstavlja vremenski zahtevan deo procesa

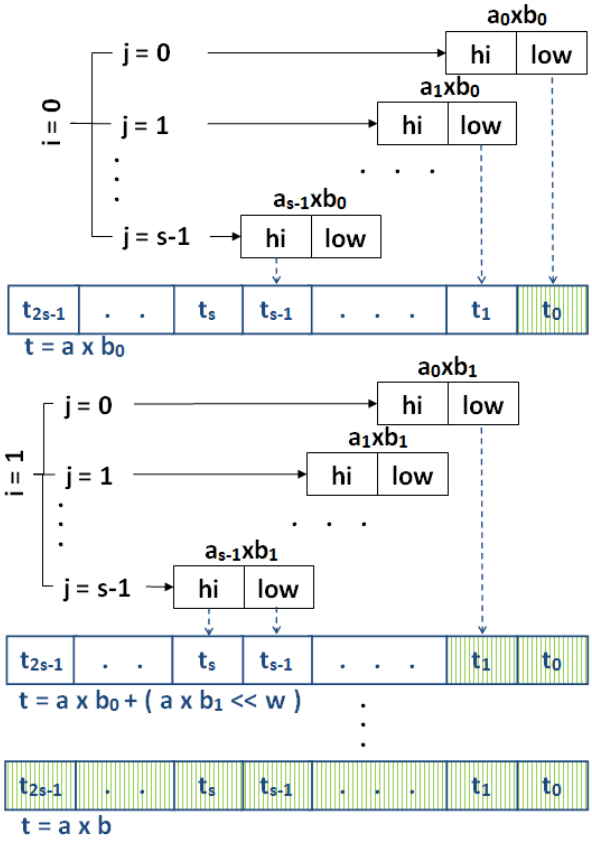
enkripcije i vrši procesiranje velikih celobrojnih vrednosti, realizuje se u nekoliko koraka, pri čemu je prvi korak dat listingom na slici 5.2.

```
for i = 0 to s-1
    C := 0
    for j = 0 to s-1
        (C, S) := t[i + j] +  a[j]·b[i] + C
        t[i + j] := S
    t[i + S] := C
```

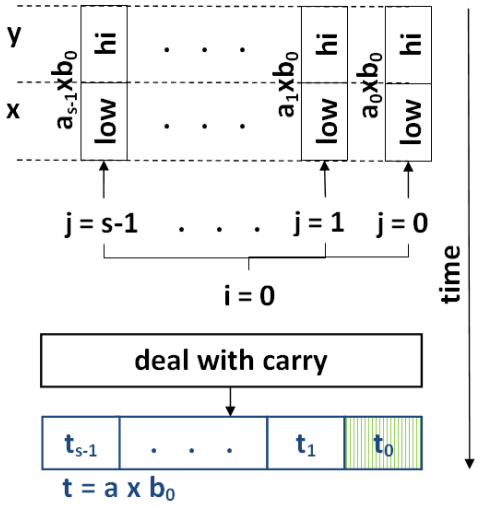
Slika 5.2 Korak 1 Montgomerijeve redukcije: množenje velikih celobrojnih vrednosti *a* i *b*, procesiranjem na nivou cifara.

Broj cifara neophodnih za predstavljanje velikih celobrojnih vrednosti *a* i *b* je na slici označen sa *s* i njegova vrednost zavisi, između ostalog, od izbora arhitekture i korišćenog matičnog tipa podataka u okviru programa. U unutrašnjoj petlji se vrši sabiranje međurezultata *t*, proizvoda *i*-te cifre broja *b* i *j*-te cifre broja *a*, kao i prenosa *C* iz prethodne iteracije.

Redosled operacija na nivou cifara dugačkih brojeva *a* i *b* sa slike 5.2 se može ilustrovati kao što je opisano u radu [53] i prikazano slikom 5.3. Postavka na slici 5.3(b) je korišćena u sprovedenim Maxeler testovima, pri čemu se razlika ogleda u redosledu procesiranja prenosa u odnosu na slučaj sa slike 5.3(a). U slučaju (a) prenos se računa pri svakom množenju dve cifre, odnosno za svaku vrednost promenljive *j*. U slučaju (b), prvo se vrši množenje svih cifara, bez dodavanja prenosa, ali se prenos pamti i dodaje na međurezultat tek na kraju ciklusa množenja, odnosno po izlasku iz petlje po promenljivoj *j*. Izbegavanjem procesiranja prenosa iz prethodne iteracije, eliminišu se ciklični grafovi u hardveru, ali se kao posledica javlja potreba za dodatnim procesiranjem na strani host procesora.



(a)



(b)

Slika 5.3 Prilagođenje algoritma dataflow arhitekturi: (a) Korak 1 Montgomerijeve redukcije sa procesiranjem na nivou cifara, gde je parametar *w* širina cifre (broj bita); (b) Ažuriranje prenosa (*carry*) na kraju ciklusa množenja, umesto ažuriranja na kraju svake iteracije.

Ubrzanje ostvareno prilikom množenja, prikazano je tabelom 5.1. Pritom, kašnjenje prilikom procesiranja prenosa na strani host procesora nije uzeto u obzir. Kernel je realizovan prema modelu sa slike 5.3(b), pri čemu se u hardveru dobija množač koji prihvata dva niza cifara za brojeve *a* i *b*, tako da je širina cifre 32 bita. U tom slučaju se prilikom množenja dve 32-bitne cifre dobija 64-bitni rezultat podeljen u 32-bitni niži razred i 32-bitni viši razred, koji su označeni na slici sa *low* i *hi*, respektivno (nizovi podataka *x* i *y*). Ovakvi nizovi se zatim šalju ka host strani gde se formira međurezultat *t*.

Međutim, pretpostavka je da i ovo kašnjenje može biti redukovano uz upotrebu Maxeler podrške za kontrolu toka izvršavanja programa u vidu dinamičkih ofseta, brojača i kontrolnih nizova [54]. RSA je blok algoritam, što omogućava enkripciju dugačkih tokova podataka koji se mogu posmatrati kao skupovi blokova. Ta činjenica je iskorišćena za realizaciju ulaznog opterećenja za koje se može dobiti značajno ubrzanje. Maksimalno postignuto ubrzanje na osnovu tabele iznosi 69% i odnosi se na operaciju množenja u okviru prvog koraka Montgomerijevog algoritma datog slikom 5.2.

Tabela 5.1. Tabela ubrzanja za različit broj DFE množača, u odnosu na standardnu CPU arhitekturu.

VELIČINA ULAZNOG SETA [MB]	UBRZANJE [%]	BROJ MNOŽAČA (*LANES*)
1	**31**	**2**
4	**55**	**2**
40	**56**	**2**
400	**61**	**2**
1	**31**	**4**
4	**57**	**4**
40	**64**	**4**
400	**69**	**4**

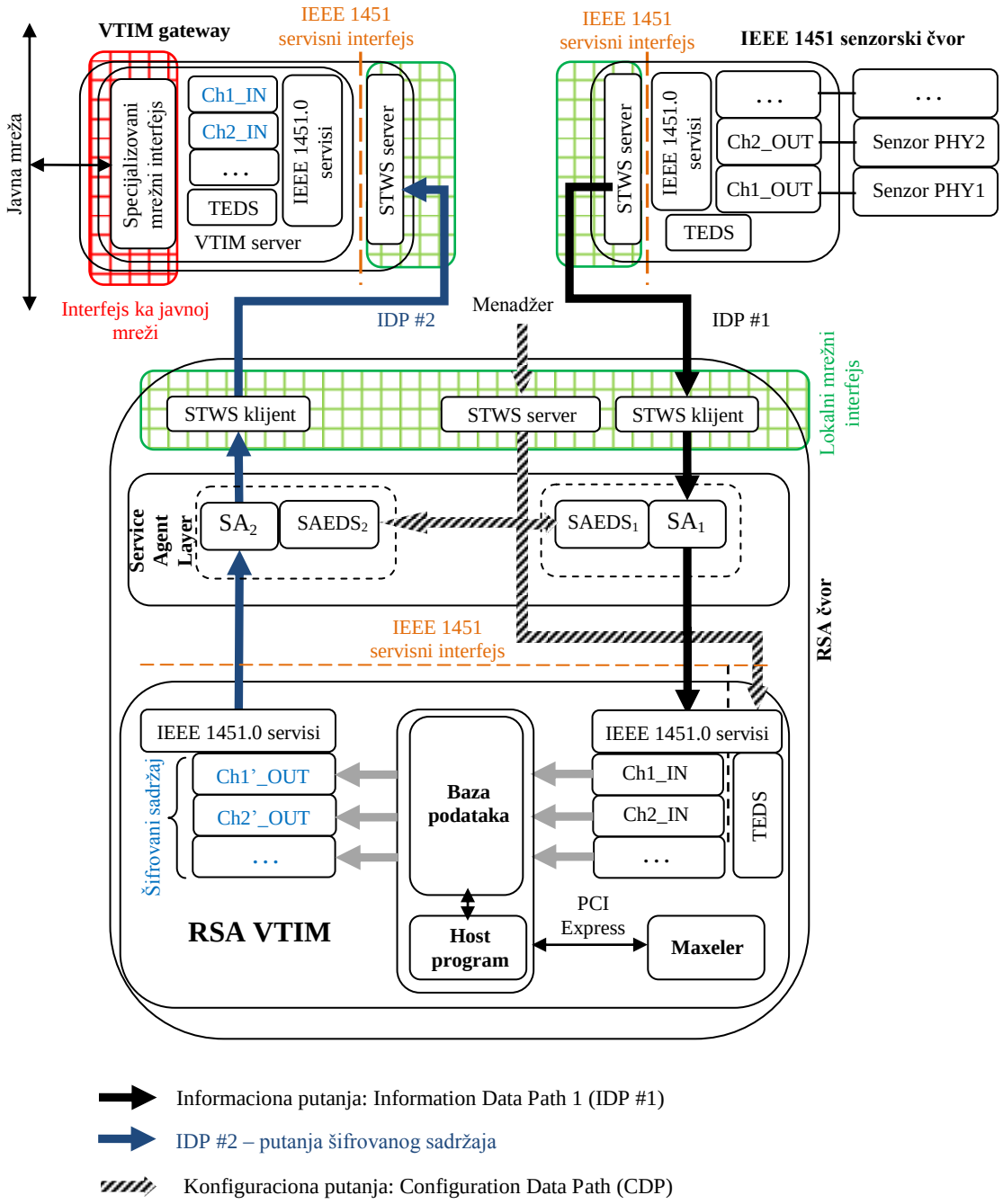
Ubrzanje od 69% je značajno, ali funkcija koja je izmeštena na DFE obuhvata svega 40% od ukupnog vremena izvršavanja programa na CPU platformi. Na osnovu Amdalovog zakona, dolazi se do ukupnog ubrzanja od 28%. Za dalja poboljšanja, moguće je koristiti istovremenu obradu na host strani i na DFE strani.

Mrežna integracija Maxeler čvora predstavlja važan aspekt, jer izvorne i šifrovane podatke treba razmenjivati u mreži na efikasan način. Uvođenje servisno orijentisane arhitekture unapređuje interoperabilnost u mreži i omogućava da se *dataflow* platforma integriše u vidu servisa za enkripciju podataka. Za potrebe mrežne integracije, Maxeler platformi se može pristupiti preko STWS servisa gde se odgovarajući formati definišu WSDL specifikacijom, čime se postiže implementacija Maxeler mrežnog čvora u VTIM formi. Koncept je prikazan slikom 5.4.

Menadžerski čvor konfiguriše nezavisne putanje podataka u lokalnoj mreži, slično kao u slučaju implementacije GPC algoritma u mreži pametnih pretvarača. Prema konceptualnom prikazu, servisni agent SA_1 vrši prenos podataka iz više mernih kanala, nakon čega se oni smeštaju u bazu na strani RSA čvora. Na taj način, stvara se kolekcija podataka prikupljena sa različitih senzora u dužem vremenskom intervalu, što predstavlja odgovarajuće ulazno opterećenje za Maxeler. Host program preuzima podatke odgovarajućeg kanala iz baze podataka, šalje ih na FPGA akcelerator i konačan rezultat upisuje natrag u bazu. Nakon konfiguracije putanje šifrovanih podataka IDP #2, šifrovani sadržaj se čita iz izlaznih kanala (Ch1'_OUT, Ch2'_OUT, ...) koji su mapirani ka podacima baze i upisuje periodično u *gateway* čvor realizovan takođe u VTIM formi. VTIM kao *gateway* obezbeđuje interfejs ka javnoj mreži koji može biti proizvoljan i koji obezbeđuje sigurnosnu razmenu podataka sa udaljenim klijentom. Pritom, parametri enkripcije poput broja cifara s, širine cifre w, dužine ključa i veličine bloka koji se učitava od strane host programa iz baze podataka, kao i adekvatne opise šifrovanog sadržaja, treba specificirati preko posebno definisanih TEDS, što neće biti dalje razmatrano.

Integracija *dataflow* platforme u VTIM formi omogućava brzo šifrovanje velike količine mernih podataka u mreži pametnih pretvarača, pri čemu se prikupljanje podataka vrši upotrebom servisnih agenata. Maxeler platforma je pogodna za obradu velike količine ulaznih podataka, ali ne rešava problem kreiranja takvog ulaznog seta. Upotrebom VTIM modela, rešava se problem

prikupljanja ulaznih podataka i njihovog prevođenja u format pogodan za Maxeler obradu.



Slika 5.4 Koncept integracije Maxeler platforme u VTIM formi za potrebe ubrzanja procesa enkripcije podataka u lokalnoj servisno orijentisanoj mreži pametnih pretvarača.

Povezivanje Maxeler čvora i pametnih pretvarača se u lokalnoj mreži vrši konfiguracijom na standardan način, putem GUI interfejsa menadžerskog čvora, tako da se formiraju putanje „sirovih“ i putanje šifrovanih podataka. Još jedan koristan vid upotrebe VTIM modela je dat prilikom uvođenja *gateway* čvora, čime se postiže enkapsulacija poslova lokalne mreže i obezbeđuje servis za čitanje šifrovanog sadržaja u okviru javne mreže. VTIM model ne postavlja platformska ograničenja, pa je primenljiv u različitim scenarijim korišćenja *multi-core*, *many-core* i *dataflow* arhitektura.

5.2. Servisni agenti na bazi SNMP protokola

Fleksibilnost predloženog modela implementacije se ne ogleda samo u korišćenju različitih arhitektura uređaja prilikom VTIM implementacije, već je moguće i koristiti različite komunikacione protokole na nezavisnim putanjama podataka. U prethodno navedenim primerima je korišćen HTTP protokol za razmenu SOAP/XML poruka, čime se postiže visok nivo interoperabilnosti između mrežnih čvorova, ali se mogu javiti i aplikacije koje zahtevaju žrtvovanje interoperabilnosti radi boljih performansi. Zbog toga je u nastavku dat primer upotrebe SNMP protokola za pristup IEEE 1451.0 operacijama, koji služi kao osnova za realizaciju alternativnog protokola servisnog agenta.

SNMP je popularni protokol koji se koristi za Internet Protocol (IP) mrežni menadžment, odnosno za prikupljanje podataka sa mrežnih uređaja poput servera, hub uređaja, štampača i rutera, kao i za njihovu konfiguraciju. Razvijen je kao podrška monitoringu mrežnih uređaja 1988. godine, a 1990. je proglašen Internet standardom od strane Internet Architecture Board (IAB), od kada je u širokoj upotrebi. U nastavku je dat opis industrijskih standarda vezanih za upotrebu SNMP verzije 1 i 2c, a na osnovu Requests for Comments (RFC) dokumenata [55-57]:

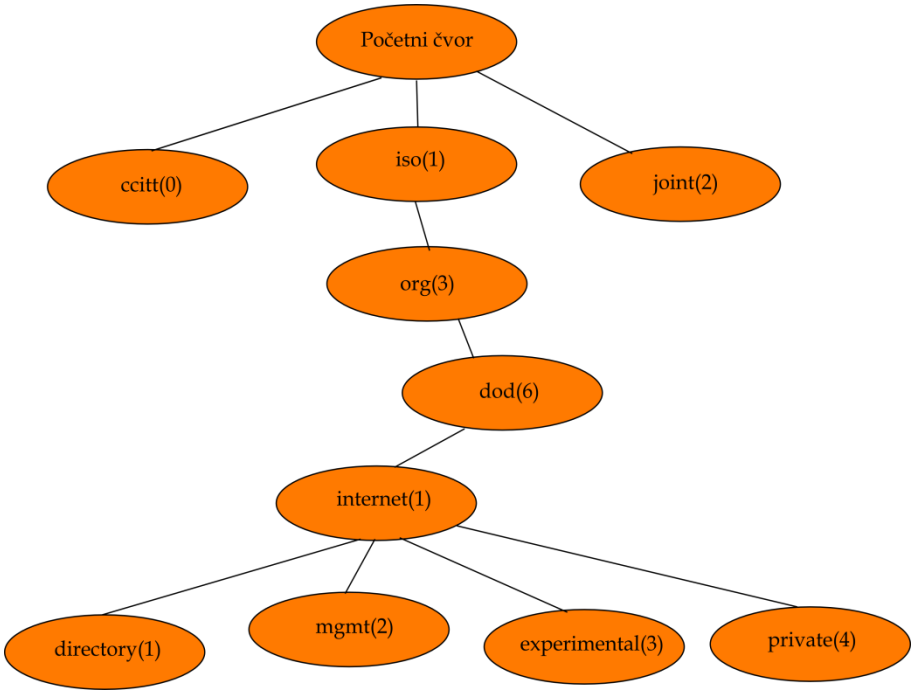
- **SNMP** - RFC 1157, “A Simple Network Management Protocol (SNMP)”, standard koji definiše proces komunikacije među SNMP-podržanim uređajima i formate poruka koje se tom prilikom razmenjuju. SNMPv2c je verzija definisana u okviru RFC 1901, RFC 1905 i RFC 1906. SNMP tipično koristi User Datagram Protocol (UDP) kao transportni protokol.

- **Management Information Base (MIB)** - RFC 1213, "Management Information Base for Network Management of TCP/IP based Internets: MIB II." Ovim standardom se definiše set objekata kojima se predstavljaju informacije pojedinih IP čvorova mreže, kao što je lista mrežnih interfejsa, tabela rutiranja, Address Resolution Protocol (ARP) tabela, lista otvorenih TCP konekcija, ili Internet Control Message Protocol (ICMP) statistika. Standard je ažuriran uvođenjem RFC 2011-2013, čime se podrazumeva i rad na bazi UDP protokola.
- **Structure of Management Information (SMI)** - RFC 2578, "Structure of Management Information Version 2 (SMIv2)." Standard koji definiše objektnu sintaksu za specificiranje kako se MIB podaci referenciraju i čuvaju.

Arhitektura SNMP sistema je podeljena na dva dela: menadžerski i agentski. Agent predstavlja program koji se izvršava na udaljenom uređaju koji se nadzire i konfiguriše. Na zahtev menadžera, agent očitava ili modifikuje stanje MIB tabele koja se nalazi na uređaju, pri čemu i agent može da inicira komunikaciju pomoću *trap/notification* mehanizma, odnosno u alarmnim situacijama. U radu [58], dat je opis implementacije skalabilne mreže pametnih pretvarača uvođenjem TEDS sadržaja putem MIB tabele kojoj se pristupa preko SNMP agenta. Prikazan je poseban format MIB tabele i korišćenje gotovih komponenti za realizaciju pametnog pretvarača. Primeri realizacije plug & play koncepta i standardizacije MIB tabele na osnovu IEEE 1451.4 standarda dati su u [59]. Time je, između ostalog, dat način standardizacije aplikacionog sloja zaduženog za obavljanje komunikacije u mreži pametnih pretvarača. U radu [60], autori predviđaju upotrebu IEEE 1451.0 i SNMP protokola, što je zapravo cilj eksperimenta datog u nastavku. Konkretno, opisan je primer upotrebe SNMP protokola kao zamene za HTTP API koji je definisan standardom IEEE 1451.0.

U okviru SNMP protokola, MIB tabeli uređaja koji se nadzire ili konfiguriše, pristupa se pomoću Object ID (OID) vrednosti. Objekti kojima se predstavljaju statusne informacije uređaja i kojima upravlja SNMP agent, organizovani su putem hijerarhije u vidu stabla [61] (slika 5.5). Na taj način svaki čvor stabla ima svoj OID koji se sastoji iz niza celobrojnih vrednosti sa

tačkom kao separatorom. Prema datoj slici, ukoliko se pristupa internet podstablu, potrebno je koristiti putanju *iso(1).org(3).dod(6).internet(1)*, kojoj odgovara OID u formi celobrojnih vrednosti 1.3.6.1 ili u formi razumljivijoj za korisnika: *iso.org.dod.internet*. Pored unapred definisanih OID vrednosti u okviru standarda, ostavljena je mogućnost za registraciju privatnih OID vrednosti, koje se zatim koriste za specifičnu upotrebu. Za tu svrhu, rezervisano je podstablo *iso.org.dod.internet.private.enterprise* (1.3.6.1.4.1), gde se privatni brojevi registruju pomoću centralnog autoriteta *Internet Assigned Numbers Authority* (IANA; *http://www.iana.org/*). Na taj način je za NIST institut rezervisan broj 724, odnosno podstablo 1.3.6.1.4.1.724. Neki od rezervisanih brojeva su takođe: 11067- Univerzitet u Beogradu, 2-IBM, 3-Carnegie Mellon i 9-Cisco.



Slika 5.5 Čvorovi SNMP stabla definisani SMI specifikacijom.

Sa obzirom na to da će ovde biti dat primer OID vrednosti koje nisu definisane standardom, za pristup MIB tabeli pametnog pretvarača biće korišćeno fiktivno podstablo 1.3.6.1.4.1.50000.1451.0, odnosno *iso.org.dod.internet.private.enterprises.soa.1451.0*. Broj 50000 je izabran jer je

slobodan prema trenutnom stanju IANA registra. Po uzoru na 1451.0 HTTP API definiciju opisanu u sekciji 2.2, u nastavku će biti opisan primer definicije jednog dela SNMP API-ja datog tabelom 5.2.

Parametar <path> je HTTP URL putanja definisana u okviru 1451.0 HTTP API-ija (tabela 2.3) i jednoznačno određuje NCAP komandu koja se poziva. Za definiciju iste komande u SNMP domenu, potrebno je konvertovati HTTP URL putanju u SNMP OID niz (kolone 2 i 3). Parametar <snmp_path_txt> iz tabele 5.2 predstavlja deo OID niza kojim se NCAP komanda jednoznačno određuje u SNMP domenu. Parametri <snmp_path_txt> i <snmp_path_num> određuju isti OID, sa tom razlikom što se prvi koristi u tekstualnoj formi OID-a, a drugi u numeričkoj formi. Konačan OID za slanje pametnom pretvaraču, dobija se prema sledećoj sintaksi: <prefix>.<snmp_path_txt>.<snmp_param>, tako da prefiksni parametar <prefix> ima sledeću vrednost:

<prefix> = "*iso.org.dod.internet.private.enterprises.soa.1451.0*",

a <snmp_param> je niz celobrojnih vrednosti koji se šalju kao parametri IEEE 1451.0 komande.

Tabela 5.2. Konverzija IEEE 1451 HTTP URL-a u OID, u cilju pristupanja IEEE 1451.0 komandama putem SNMP protokola.

<path>	*<snmp_path_txt>*	*<snmp_path_num>*
1451/TEDSManager/ReadTeds	TEDSManager.ReadTeds	1.1
1451/TEDSManager/ReadRawTeds	TEDSManager.ReadRawTeds	1.2
1451/TEDSManager/WriteTeds	TEDSManager.WriteTeds	1.3
1451/Discovery/TIMDiscovery	Discovery.TIMDiscovery	2.1
1451/Discovery/TransducerDiscovery	Discovery.TransducerDiscovery	2.2
1451/TransducerAccess/ReadData	TransducerAccess.ReadData	3.1
1451/TransducerAccess/WriteData	TransducerAccess.WriteData	3.4

Prema prikazanom modelu, svakoj operaciji pametnog pretvarača, zajedno sa parametrima koji se prosleđuju, pridružen je tačno jedan SNMP OID. Iz tabele se vidi da se svakom interfejsu 1451.0 API-ja dodeljuje po jedan identifikacioni broj (broj pre tačke u koloni 3). Identifikacioni brojevi 1, 2 i 3 dodeljeni su interfejsima TEDSManager, Discovery i TransducerAccess, respektivno. Takođe, svaka metoda koja pripada određenom interfejsu, označena je celobrojnom vrednošću koja se navodi posle tačke u trećoj koloni tabele. Kako bi poziv komande bio kompletan, potrebno je u OID niz uvrstiti i parametre koji se šalju određenoj komandi. Na primer, ukoliko se pristupa IEEE 1451.0 komandi readTeds za čitanje sadržaja tipa MetaTEDS: potrebno je pozvati se na prvu vrstu iz tabele 5.2 i dopuniti OID niz parametrom čija vrednost prestavlja MetaTEDS pristupni kod. Takođe, potrebno je proslediti i parametar čija vrednost definiše timeout period za čitanje sadržaja. U tom slučaju se dobija sledeća OID vrednost: <prefix>.1.1.1.5, gde su poslednje dve cifre parametri koji se prosleđuju (dopuna početnog OID-a), a prve dve cifre operacija readTEDS (početni OID iz tabele 5.2). Pristupni kod za MetaTEDS sadržaj je 1, dok je timeout vrednost postavljena na 5s. Dodavanjem numeričkog prefiksa, dobija se kompletna numerička predstava ovog OID-a: 1.3.6.1.4.1.50000.1451.0.1.1.1.5. Mana ovog pristupa jeste što se za različite vrednosti timeout-a dobijaju različite OID vrednosti, pa se samim tim vrši i pristup različitim MIB resursima. Praktično, resurs kome se pristupa je samo jedan, i to je deo memorije koji sadrži MetaTEDS podatke. Jednostavnija realizacija se može dobiti ukoliko se timeout vrednost zada fiksno u okviru NCAP-a i taj parametar izbaci iz OID niza. U suprotnom, SNMP agent koji se izvršava na NCAP procesoru mora biti u stanju da parsira OID niz, izdvoji timeout parametar i koristi ga kao argument metode.

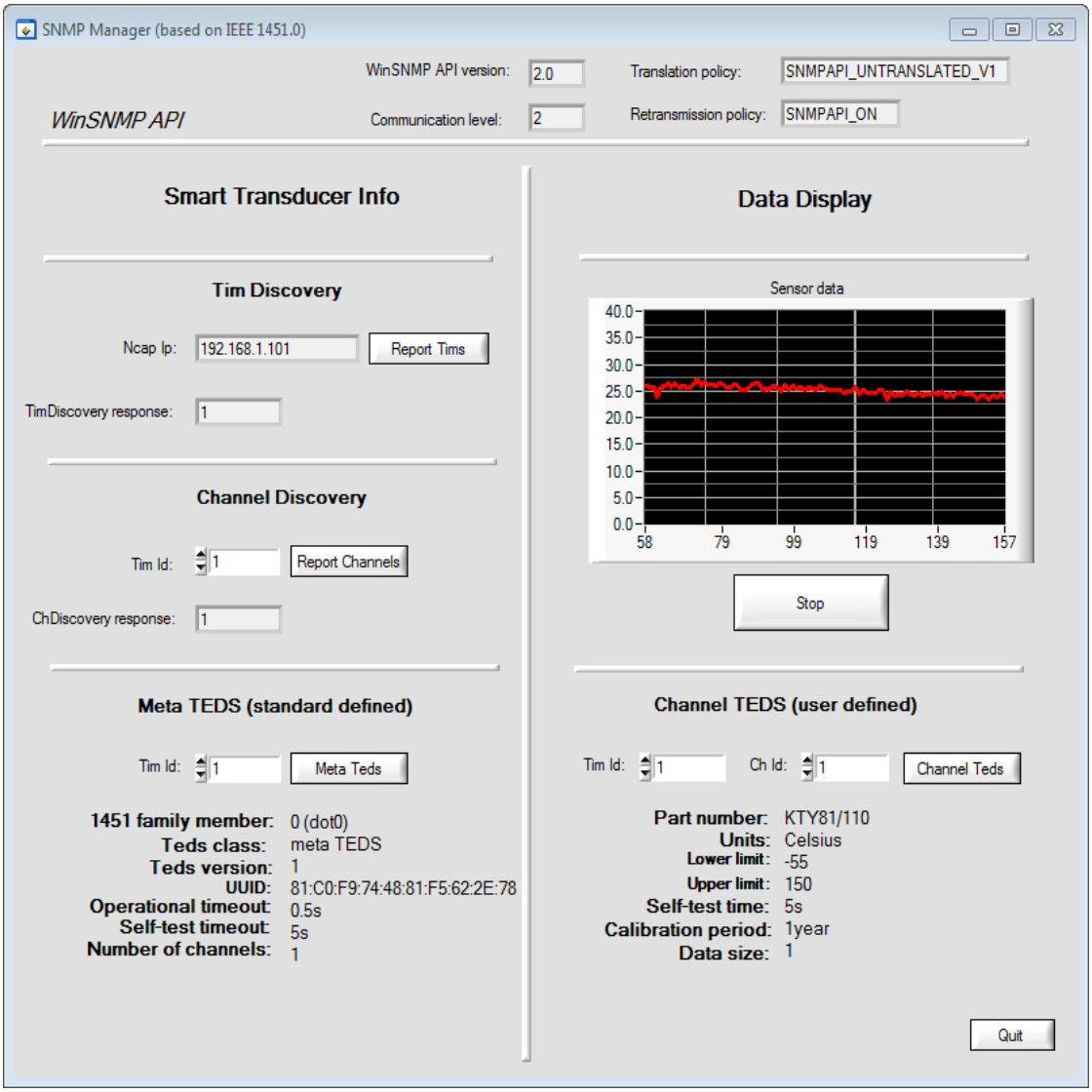
U prikazanoj realizaciji, MIB komponente su zapravo 1451.0 metode sa predefinisanim parametrima. U tom slučaju se pristupa celokupnom sadržaju koji se dobija kao povratni argument pozivane metode. Međutim, moguće je razmotriti i druge realizacije u kojima se pristupa samo delu pomenutog

sadržaja. Na primer, ukoliko je potrebno pročitati samo jedno polje iz MetaTEDS-a, a ne ceo sadržaj, potrebno je definisati OID nizove koji omogućavaju čitanje sa većom rezolucijom.

Prototip generičkog modula pametnog pretvarača koji je opisan u sekciji 3.1 je izmenjen u cilju realizacije namenskog SNMP agenta. SNMP agent je realizovan na istom mikrokontroleru tako što je modifikovana već postojeća SNMP biblioteka. Sa novodobijenim modulom su povezani temperaturni senzor za merenje temperature vazduha, kao i ruter za potrebe testiranja komunikacije u LAN mreži. Za potrebe testiranja implementirane su sledeće metode iz tabele 5.2: *TIMDiscovery*, *TransducerDiscovery*, *ReadData* i *ReadTeds*. Takođe, pojednostavljena je realizacija TEDS komponenti. MetaTEDS podaci su realizovani u vidu niza ASCII karaktera, gde su različita TEDS polja razdvojena praznim znakom i to u sledećem formatu [2]: "<*1451 family member*> <*Teds Class*> <*Teds version*> <*UUID*> <*Operational timeout*> <*Self-test timeout*> <*Number of channels*>". Prilikom korišćenja HTTP API-ja, ovakav ASCII niz bi predstavljao tekstualno formatiran HTTP odgovor, koji se tipično dobija konverzijom iz binarnog TEDS sadržaja.

Za realizaciju menadžerskog čvora, korišćeno je programsko okruženje NI LabWindows CVI. Pristup udaljenom SNMP pametnom pretvaraču je ostvaren preko Win32 API-ja, koji je programeru dostupan u projektu u vidu biblioteke. U LabWindows CVI okruženju, kreiran je korisnički interfejs za prikupljanje TEDS podataka, što je funkcija menadžerskog čvora. Sa druge strane, u okviru istog interfejsa, kreiran je i displej za demonstraciju VTIM funkcionalnosti. Servisni agent je realizovan upotrebom tajmera sa pridruženom callback funkcijom. Ova callback funkcija ima za cilj da pošalje SNMP zahtev agentu (pametnom pretvaraču), za šta se oslanja na odgovarajuće WinSNMP API funkcije. Takođe, u okviru korisničkog interfejsa, implementirane su kontrole kojima su pridružene callback funkcije za sledeće četiri 1451.0 komande iz tabele 5.2: *TIMDiscovery*, *TransducerDiscovery*, *ReadTeds* i *ReadData*. Prikaz glavnog prozora programa, dat je na slici 5.6. U zaglavlju

prozora su ispisani podaci o verziji i režimu rada Microsoft WinSNMP implementacije. Na levoj strani su komande za slanje SNMP zahteva prema prototipu pametnog pretvarača za dobijanje osnovnih podataka o modulu. Na taj način se redom zahtevaju: lista dostupnih TIM modula, lista dostupnih pretvaračkih kanala (za navedeni TIM modul) i MetaTEDS podaci (za navedeni TIM modul). U prikazanom slučaju, dostupan je samo jedan kanal (senzor temperature).



Slika 5.6 Prikaz glavnog prozora menadžer/VTIM čvora realizovanog na bazi SNMP protokola.

Za informacije o kanalu, potrebno je pritisnuti dugme "*Channel Teds*", pri čemu se učitava i prikazuje jedan segment TEDS sadržaja kanala. Konačno, posle prikupljanja pomenutih informacija, startuje se servisni agent koji učitava vrednosti senzora sa periodom od 200ms, što je perioda pozivanja *callback* funkcije tajmera.

Polazeći od specifikacije IEEE 1451.0 standarda u kojoj se za pristup pametnim pretvaračima koristi HTTP protokol, predstavljen je još jedan način realizacije pristupa korišćenjem SNMP protokola. Opisana je konverzija standardnih HTTP putanja koje su date u URL formatu, u OID identifikatore SNMP protokola. Zatim je opisan prototip SNMP menadžera i servisnog agenta korišćenjem dostupnih biblioteka i prikazani su rezultati testiranja komunikacije. Time je demonstrirana još jedna mogućnost realizacije mrežnih komponenata prema predloženom servisno orijentisanom modelu u kome je razmena podataka pod kontrolom servisnih agenata bez obzira na izabrani komunikacioni protokol. Na taj način je moguća i realizacija hibridnih mreža u kojima mrežni čvorovi daju specifikacije komunikacionih sposobnosti kroz posebno definisane TEDS strukture. U tom slučaju, operater koristi SAEDS strukture za specifikaciju putanja podataka koji se razmenjuju između kanala sa kompatibilnim komunikacionim protokolima.

5.3. Implementacija modela u opštem slučaju

Model koji se zasniva na upotrebi servisnih agenata, implementiran u okviru studije slučaja distribuiranog upravljanja i opsian u sekciji 4.2.2, pruža mogućnost implementacije na različitim platformama, kako namenskim, tako i na platformama opšte namene. Pritom, implementacija modela zahteva upotrebu objekata za sinhronizaciju, ali sam model ne definiše skup konkretnih objekata koje treba upotrebiti. Programer bira konkretne objekte u formi muteksa, semafora, uslovnih varijabli i slično, prema izabranoj platformi, kako bi ostvario funkcionalnosti zadate modelom.

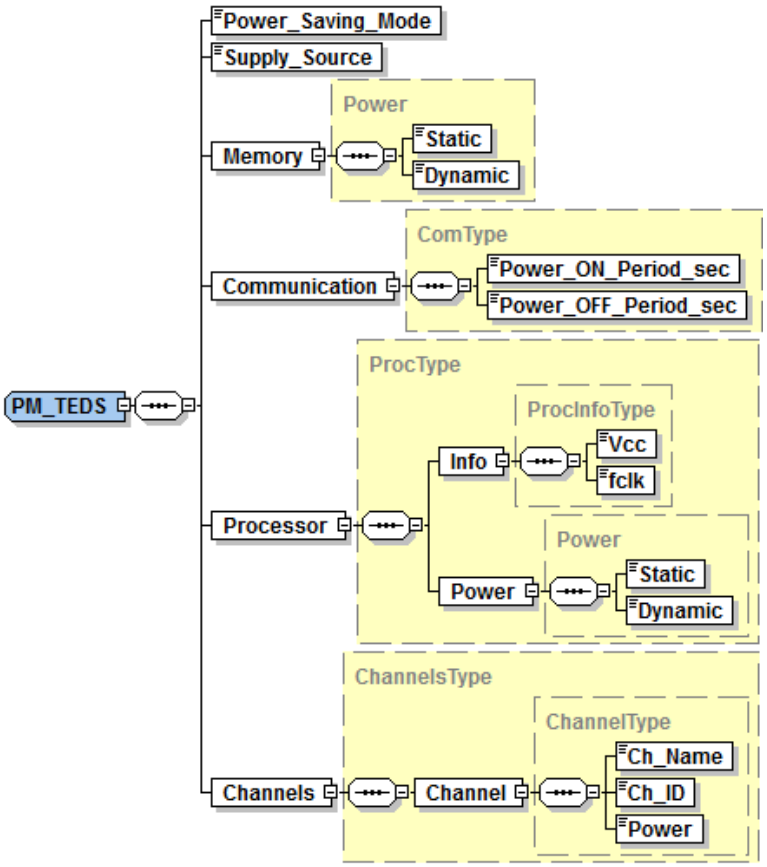
Detekcija grešaka u prikazanom modelu je data grupisanjem kodova grešaka prema slojevima koji su obuhvaćeni implementacijom određenog

mrežnog čvora. Na primeru IEEE 1451.0 interfejsa, a u slučaju da se NCAP i TIM realizuju kao zasebni moduli, može se uočiti podela kodova grešaka prema izvoru nastanka greške, tako da se javljaju četiri moguća izvora: lokalni IEEE 1451.0, lokalni IEEE 1451.X sloj, udaljeni IEEE 1451.0 i udaljeni IEEE 1451.0 sloj. NCAP tumači određene bite koda greške, kako bi dobio podatak o tome da li je on sam izvor greške ili je to ipak udaljeni TIM modul, kao i podatak o sloju koji je izvor greške: komunikacioni (1451.X) ili servisni (1451.0). Zatim se tumače preostali biti, kako bi se dobila informacija o vrsti greške (npr. nedostatak memorijskih resursa, neispravna TEDS struktura i sl.). U modelu združenog NCAP i TIM modula koji je ovde korišćen, kod greške podrazumeva uvek NCAP kao izvor greške. Status greške IEEE 1451 sloja se na jednostavan način može uvrstiti u formate poruka servisa na mrežnom sloju, pa tako i prenositi kroz mrežu putem STWS, ali drugih mrežnih servisa koji nisu STWS, u vidu celobrojne vrednosti.

Osim toga, svaki od slojeva STWS steka ima sopstvenu grupu kodova grešaka. Na primer, HTTP greške detektuje i procesira HTTP server korišćen u konkretnoj implementaciji. Dizajneru konkretnog sistema se ostavlja sloboda izbora prilikom upotrebe HTTP servera/klijenta koji su platformski zavisni. Slično kao u slučaju detekcije greške, sigurnost podataka zavisi od konkretne implementacije datog modela, pri čemu se dizajner sistema može osloniti na gotova rešenja poput sigurnosnog HTTP protokola i SOAP sigurnosne podrške [62].

Potrošnja je još jedan aspekt implementacije mreže, posebno značajan za realizaciju bežičnih mrežnih čvorova sa baterijskim napajanjem. Iz ovog aspekta, javlja se platformska zavisnost, tako da potrošnja jako zavisi od izbora komponenata za realizaciju centralnog procesora i komunikacionog interfejsa. IEEE 1451.0 definiše posebnu *TIM Sleep* komandu za optimizaciju potrošnje bežičnih TIM-ova, ali bez detalja same realizacije. Kako bi se ostvarila preciznija kontrola potrošnje u radu [63] je definisan poseban TEDS format (slika 5.7). Upisom vrednosti *ON* u polje *Power_Saving_Mode* dozvoljava se primena

tehnika uštede energije koje uređaj podržava, pri čemu ostala polja definišu režim rada pojedinih domena (delova uređaja). Inicijalno, ovo polje ima vrednost *OFF* i uređaj radi u aktivnom režimu sa punom potrošnjom, kako bi svi servisi bili aktivni, pre nego što se završi proces konfiguracije mreže pomoću centralnog menadžera. Snaga disipacije memorije i procesora se može podeliti na statičku i dinamičku, pri čemu polja *Static* i *Dynamic* definišu aktivaciju/deaktivaciju ovih komponenti snage.



Slika 5.7 XML Schema dijagram TEDS strukture za definisanje energetskog TIM profila.

Praktično, dinamička snaga se može eliminisati ukidanjem signala takta ciljanog modula, dok se statička snaga može eliminisati ukidanjem napajanja modula. Prema tome, u slučaju da se zahteva brži prelaz iz neaktivnog u aktivni režim, predviđeno je ukidanje dinamičke komponente. Kao primer ukidanja statičke komponente i sporijeg vraćanja u aktivni režim rada, može se navesti

ukidanje napona oscilatora mikrokontrolera i startovanje oscilatora po detekciji odgovarajućeg prekida. Snaga pojedinačnih kanala se može uštedeti ukidanjem napona AD/DA konvertora, kao i ukidanjem napona kola za obradu signala senzora/aktuatora. Parametri polja za komunikaciju obuhvataju vremenske intervale u kojima je RF primopredajnik ili neki drugi komunikacioni interfejs aktivan/neaktivan. Informacije o tipu izabranog napajanja, kao i dodatne informacije o napajanju (npr. stanje baterije), sadržane su u polju *Supply_Source*. U slučaju da je potrebno ostvariti veći nivo kontrole, moguće je u postojeće domene uvesti dodatna polja, poput polja za kontrolu potrošnje primenom dinamičke promene napona i frekvencije kao u [64]. I u ovom slučaju, od konkretne implementacije zavisi da li će mrežni čvor podržati dati TEDS format i na koji način će biti primenjene tehnike uštede statičke i dinamičke komponente snage. Pregled tehnika potrošnje na nivou arhitekture komponenata uređaja, dat je u radu [65].

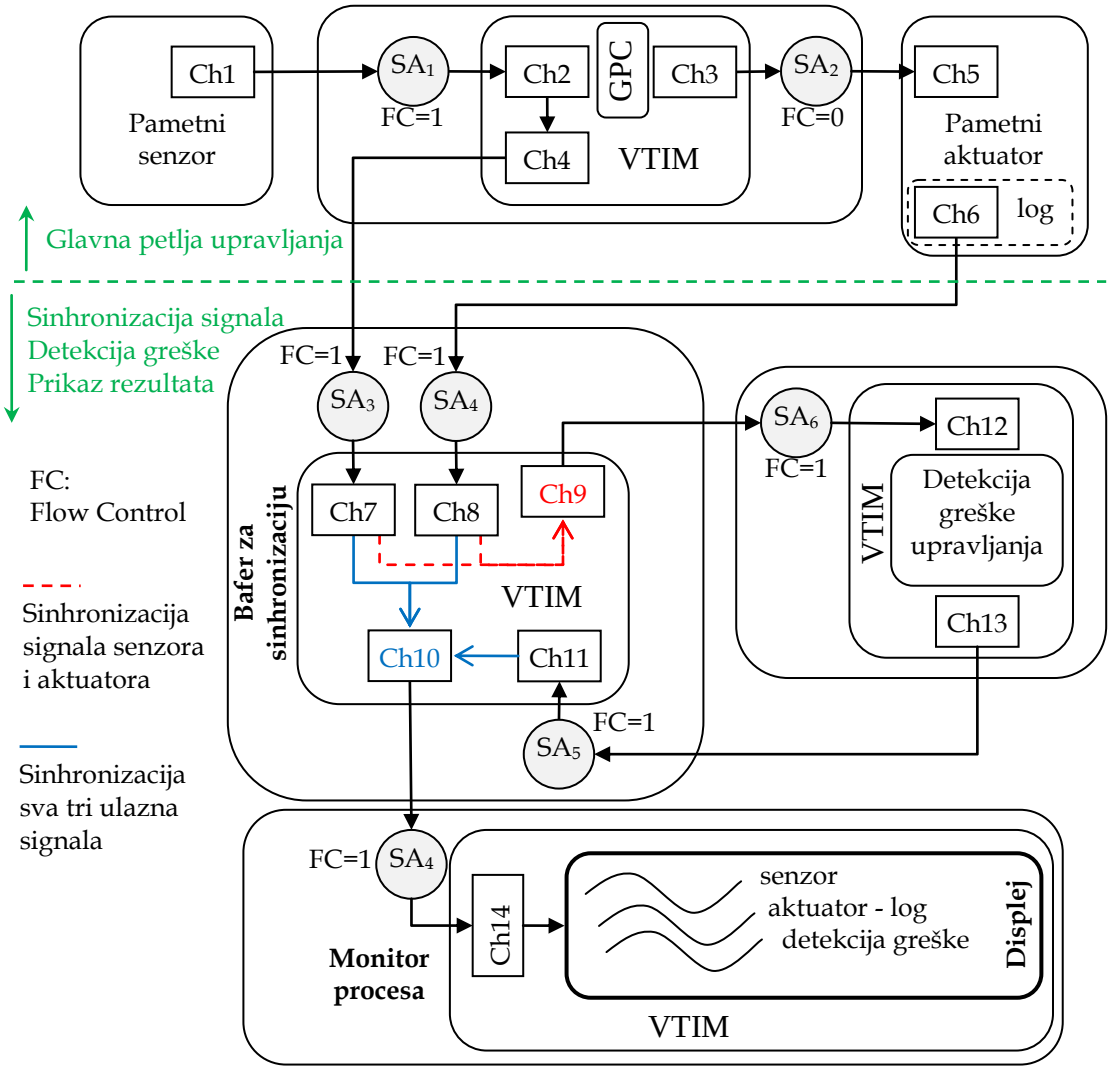
5.4. Mogućnost integracije novog modela u „Cloud Computing“ i „Internet of Things“ koncepte

Unapređivanje tehnoloških kapaciteta dovelo je do pojave koncepta računarstva u oblaku (Cloud Computing) i Interneta stvari (Internet of Things-IoT). Virtuelizacija predložena VTIM modelom daje podršku za integraciju predložene arhitekture u koncept *cloud* servisa za pristup pametnim pretvaračima. Za potrebe pristupa senzorskim kanalima, moguće je uvesti posrednički VTIM sloj u okviru *cloud*-a koji će preuzeti na sebe poslove komunikacije sa fizičkim uređajima, sa punim pristupom mernim i meta-podacima senzora. U takvoj konstelaciji, moguće je obezbediti po jednu VTIM programsku nit na strani servera za jedan uređaj ili za grupu fizičkih uređaja i to uz upotrebu postojećeg interfejsa za pristup fizičkim pametnim pretvaračima u lokalnoj mreži i proizvoljnog korisničkog interfejsa za pristup u javnoj mreži.

Jedan primer virtuelizacije podrazumeva izdvajanje grupe pametnih pretvarača sa agregiranjem njihovih kanala u jednu VTIM komponentu. Ovakav primer upravljanja pretvaračkim resursima biće dat u nastavku, kroz

primer koncepta realizacije bafera za sinhronizaciju u distribuiranom upravljačkom sistemu predstavljenom u sekciji 4.2.2.

Na slici 5.8 je data skica distribuiranog merno-upravljačkog sistema koji kao i ranije obuhvata GPC algoritam, ali koji pored glavne petlje upravljanja obuhvata i sporedne mrežne čvorove za sinhronizaciju podataka i detekciju greške upravljanja. Dodavanje sporednih čvorova ima minimalan uticaj na rad glavne kontrolne petlje, sa obzirom na to da su obezbeđeni posebni baferi interakcije između pomenuta dva sloja.



Slika 5.8 Mrežna integracija bafera za sinhronizaciju, algoritma detekcije greške i monitora procesa sa minimalnim uticajem na glavnu kontrolnu petlju.

U okviru glavne kontrolne petlje servisni agent SA_1 vrši razmenu podataka između pametnog senzora i GPC algoritma, dok servisni agent SA_2 periodično vrši transfer rezultata predikcije ka pametnom aktuatoru. Uvođenjem dodatnih kanala Ch4 i Ch6 u vidu bafera, otvara se mogućnost praćenja odbiraka signala glavne kontrolne petlje povezivanjem tih kanala sa kanalima proizvoljnih sporednih mrežnih čvorova. Uvođenje bafera za sinhronizaciju omogućava agregiranje kanala grupe uređaja u jedan VTIM čvor, tako što se zapravo vrši agregacija vremenskih referenci odbiraka signala. Dakle, za svaku vremensku referencu se formira set podataka koji se sastoji iz pojedinačnih odbiraka signala različitih kanala. Na taj način se prilikom očitavanja kanala Ch9, dobijaju sinhronizovani odbirci signala senzora i aktuatora.

Proces prikupljanja odbiraka signala senzora i aktuatora, kao i odgovarajućih vremenskih referenci, se sprovodi u mreži pomoću agenata SA_3 i SA_4 koji nizove podataka isporučuju kanalima Ch7 i Ch8 VTIM sinhronizacionog bafera. Servisni agent SA_6 periodično očitava pakete sinhronizovanih odbiraka koji se mogu obrađivati algoritmom za detekciju greške. Zatim, servisni agent SA_5 očitava rezultat algoritma za detekciju greške i upisuje ga u kanal Ch11 VTIM sinhronizacionog bafera. Nakon toga, servisni agent SA_4 očitava vremenski sinhronizovane podatke senzora, aktuatora i servisa za detekciju greške i isporučuje ih čvoru za praćenje procesa koji se realizuje u formi VTIM displeja. Na taj način se obezbeđuje mesto za operatera u sistemu koji nadzire vremenske dijagrame signala procesa i rezultat algoritma za detekciju greške. Sinhronizacioni bafer se može smatrati čvorom koji uvodi nivo apstrakcije za grupu uređaja, pri čemu su klijentski čvorovi oslobođeni poslova sinhronizacije u vidu analize vremenskih nizova podataka koji se dobijaju iz apstrakovanih uređaja.

U radu [66], kao problem realizacije IoT koncepta se ističe nedostatak adekvatnih protokola i modela programiranja. Pritom se javlja potreba za unapređivanjem interoperabilnosti na nivou aplikacija, kao i potreba za

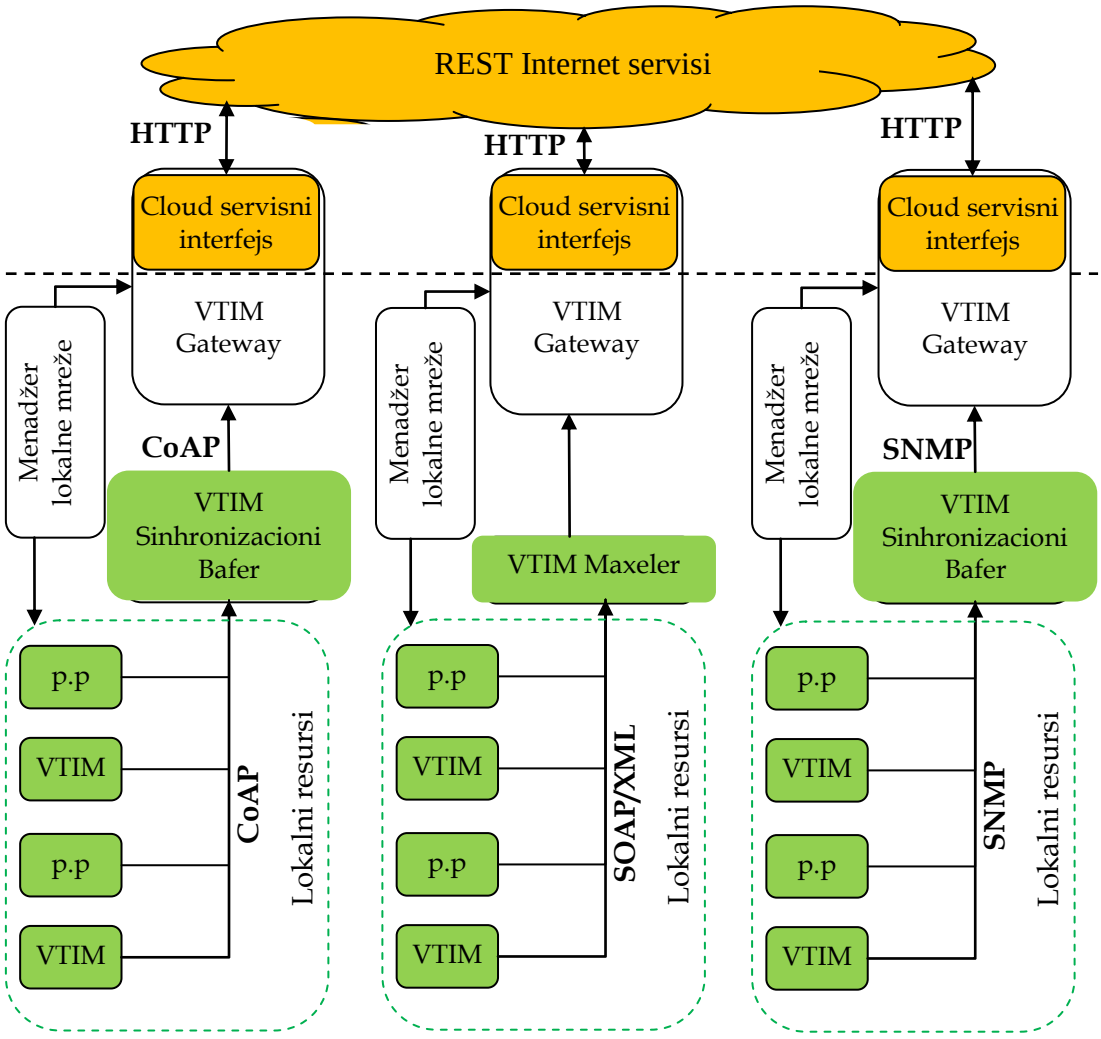
povećanjem upotrebne vrednosti uređaja za korisnike i za razvojne timove. Kao rešenje se nameće spoj tehnologija bežičnih senzorskih mreža i *World Wide Web* (WWW) tehnologije. Kao doprinos rešenju tog problema, u [66] se koristi i poseban protokol za uređaje sa ograničenim resursima: *Constrained Application Protocol* (CoAP). CoAP se zasniva na binarnom kodovanju sadržaja i za transport koristi *User Datagram Protocol* (UDP), koji je pogodniji za upotrebu u bežičnim senzorskim mrežama od TCP protokola po pitanju performansi, ali se uvodi i upotreba jednostavnog kontrolnog sloja za potrebe retransmisije paketa.

Kao što je konstatovano u [66-67], Web protokoli kao defakto rešenje za realizaciju aplikacionog sloja na Internetu, nisu uvek najpogodniji izbor pri realizaciji uređaja sa ograničenim memorijskim resursima. U generičkom prototipu koji je predstavljen u ovoj tezi, osim HTTP bafera, realizacija podrazumeva i dodatno angažovanje bafera za STWS servisni sloj što predstavlja dodatno opterećenje resursa. Međutim, arhitektura predstavljena u ovoj tezi se oslanja na koncept upotrebe VTIM mrežnog čvora za dislokaciju svih dodatnih funkcionalnosti sa namenske platforme na platformu opšte namene. Takođe, za uređaje ograničenih resursa, VTIM se može upotrebiti za realizaciju adaptera ka proizvoljnoj platformi sa proizvoljnim protokolom. Na taj način je moguće angažovati CoAP protokol, ili neki drugi protokol namenjen slabijim uređajima sa proizvoljnim modalitetom komunikacije koji se implementira kao VTIM funkcionalnost, tako da se VTIM i dalje u mreži vidi kao pametni pretvarač koji se konfiguriše posredstvom centralnog servera. Na taj način je moguće ostvariti punu interoperabilnost i omogućiti integraciju uređaja na bazi mikrokontrolera sa slabiji procesorskim i memorijskim resursima u poređenju sa korišćenim ARM mikrokontrolerom.

Iako to nije demonstrirano odgovarajućim prototipom, jasno je da se predloženi model mreže na bazi servisnih agenata može primeniti tako da servisni agenti direktno koriste CoAP protokol, čime se eliminiše potreba za VTIM konverzijom protokola. U slučaju realizacije heterogene mreže koja koristi oba protokola (i CoAP i STWS), neophodno je operatoru predočiti

mogućnosti svake komunikacione komponente definisanjem adekvatnih elektronskih specifikacija. Na taj način bi operater imao informaciju o kompatibilnim izvorišnim i odredišnim kanalima prilikom definisanja nezavisnih informacionih putanja podataka.

Pored prikazanih modela komunikacije, predlažu se i sledeći modeli integracije lokalnih resursa u javni domen (slika 5.9).



Slika 5.9 Integracija servisno orijentisane mreže u opšti model koncepta računarstva u oblaku upotrebom različitih komunikacionih protokola. U lokalnoj mreži, pametni pretvarači (p.p.) i VTIM čvorovi komuniciraju putem proizvoljno izabranog protokola.

Koncept povezivanja Maxeler platforme sa javnom mrežom je već ilustrovan slikom 5.4, dok se ovde generalizuje dati koncept i prikazuje potencijal upotrebe različitih platformi i protokola za povezivanje sa javnom mrežom i *cloud* servisima. Pritom, VTIM kao sinhronizacioni bafer predstavlja pogodnu apstrakciju koja omogućava VTIM Gateway čvoru da resurse lokalne mreže vidi kao različite stavke VTIM bafera. Menadžer konfiguriše putanje podataka u lokalnoj mreži, ali vrši i konfiguraciju javno dostupnih resursa modifikovanjem odgovarajuće elektronske specifikacije VTIM Gateway čvora. Na taj način, *cloud* servisima se može pristupiti preko sve popularnijeg *Representational State Transfer* (REST) API interfejsa, gde se pojedinim resursima dodeljuje jedinstveni identifikator resursa poznat kao *Uniform Resource Identifier* (URI). Specifikacijom URI-ja određenog VTIM-a ili pamentog pretvarača koji je javno dostupan, na *cloud* strani se mogu dobiti meta-podaci ili merni podaci datog uređaja. U okviru funkcionalnosti VTIM Gateway čvora se vrši konverzija proizvoljnog protokola na HTTP protokol u primeru sa slike. Pritom, VTIM Gateway čvor se sa stanovišta menadžera u lokalnoj mreži vidi kao i bilo koji drugi VTIM.

6. ZAKLJUČAK

U tezi je adresiran problem nedostatka standardnog modela implementacije servisno orijentisanih mreža pametnih pretvarača. Osnovni čvorovi mreže su prvobitno dati u formi centralnog servera sa aktivnom komponentom za procesiranje poslova komunikacije i pasivnim pametnim senzorima/aktuatorima i pasivnim VTIM čvorovima na kojima se izvršavaju STWS servisi. Pasivne komponente su u okviru prototipa realizovane na bazi IEEE 1451 koncepta sa nadogradnjom u vidu STWS servisa. Model je zatim unapređen uvođenjem servisnih agenata, kao aktivnih komponenata koje se mogu izvršavati na bilo kom mrežnom čvoru, čime se obezbeđuje proizvoljna topologija mreže i optimizuju performanse mrežne komunikacije.

Predloženi model implementacije na bazi procesiranja događaja rasterećuje procesorske resurse, dok podrška za vremensku sinhronizaciju omogućava prevazilaženje nepredvidljivog mrežnog kašnjenja i gubitka paketa. Tipične karakteristike mreže prema datom modelu su prikazane primerom implementacije upravljačke petlje za kontrolu temperature sušare na bazi prediktivnog algoritma upravljanja. Međutim, VTIM koncept omogućava realizaciju proizvoljne mreže, jer daje šablon za implementaciju bilo kog algoritma, ili mrežnog entiteta koji nije algoritam poput displeja za prikaz rezultata obrade i praćenje odgovarajućeg procesa. Time se pokriva širok spektar mreža koje se mogu realizovati u skalabilnoj arhitekturi, na bazi gotovih komponenata, poput mreža za kontrolu procesa, osmatranje okoline i slično. Takođe, u datoj arhitekturi moguće je korišćenje drugih servisa i funkcionalnosti koje nisu IEEE 1451, kao i drugih komunikacionih protokola poput SNMP, CoAP, UDP i HTTP protokola. Dati model je pogodan za integraciju u druge modele realizovane prema konceptima računarstva u oblaku i Interneta stvari.

LITERATURA

[1] E. Y. Song and K. Lee, "Understanding IEEE 1451-Networked smart transducer interface standard - What is a smart transducer?," *IEEE Instrumentation & Measurement Magazine,* vol. 11, no. 2, pp. 11–17, 2008.

[2] "IEEE 1451.0, Standard for a Smart Transducer Interface for Sensors and Actuators — Common Functions, Communication Protocols, and Transducer Electronic Data Sheet (TEDS) Formats," IEEE Instrumentation and Measurement Society, 2007, pp. 1-335.

[3] "IEEE 1451.2, Standard for a Smart Transducer Interface for Sensors and Actuators - Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," IEEE Instrumentation and Measurement Society, 1997.

[4] "IEEE 1451.3, Standard for a Smart Transducer Interface for Sensors and Actuators - Digital Communication and Transducer Electronic Data Sheet (TEDS) Formats for Distributed Multidrop Systems," IEEE Instrumentation and Measurement Society, 2003, pp. 1 - 175.

[5] "IEEE 1451.5, Standard for a Smart Transducer Interface for Sensors and Actuators Wireless Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," IEEE Instrumentation and Measurement Society, 2007, pp. 1 - 236.

[6] "IEEE 1451.7, Standard for Smart Transducer Interface for Sensors and Actuators-Transducers to Radio Frequency Identification (RFID) Systems Communication Protocols and Transducer Electronic Data Sheet Formats," IEEE Instrumentation and Measurement Society, 2010, pp. 1 - 99.

[7] "IEEE 1451.1, Standard for a Smart Transducer Interface for Sensors and Actuators-Network Capable Application Processor (NCAP) Information Model," IEEE Instrumentation and Measurement Society, 1999.

[8] "IEEE 1451.4, Standard for a Smart Transducer Interface for Sensors and Actuators - Mixed-Mode Communication Protocols and Transducer Electronic Data Sheet (TEDS) Formats," Instrumentation and Measurement Society, 2004, pp. 1 - 430.

[9] R. Motwani, M. Motwani, F. Harris and S. Dascalu, "Towards a scalable and interoperable global environmental sensor network using Service Oriented Architecture," in *Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2010*, Brisbane, 2010, pp. 151 - 156.

[10] M. Botts, G. Percivall, C. Reed and J. Davidson, "OGC® Sensor Web Enablement: Overview and High Level Architecture," in *GeoSensor Networks*, Springer, 2008, pp. 175-190.

[11] A. Bröring, J. Echterhoff, S. Jirka, I. Simonis, T. Everding, C. Stasch, S. Liang and R. Lemmens, "New Generation Sensor Web Enablement," *Sensors*, vol. 11, no. 3, pp. 2652-2699 , 2011.

[12] N. J. Jevtić, Automatska konfiguracija distribuiranih mernih sistema korišćenjem elektronskih specifikacija senzora, Doktorska disertacija, Elektrotehnički fakultet, Univerzitet u Beogradu, 2015.

[13] E. Y. Song and K. B. Lee, "Smart Transducer Web Services Based on the IEEE 1451.0 Standard," in *IEEE Instrumentation and Measurement Technology Conference (I2MTC)*, Warsaw, 2007, pp. 1-6.

[14] D. Wenbin, J. Peltola, V. Vyatkin and P. Cheng, "Service-oriented distributed control software design for process automation systems," in *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, San Diego, CA, 2014, pp. 3637-3642.

[15] F. Jammes, "Real time device level Service-Oriented Architectures," in *IEEE International Symposium on Industrial Electronics (ISIE)*, Gdansk, Poland, 2011, pp. 1722-1726.

[16] G. Veiga, J. N. Pires and K. Nilsson, "Experiments with service-oriented architectures for industrial robotic cells programming," *Robotics and Computer-Integrated Manufacturing*, vol. 25, p. 746–755, 2009.

[17] S. P. Lee, L. P. Chan and E. W. Lee, "Web Services Implementation Methodology for SOA Application," in *IEEE International Conference on Industrial Informatics*, Singapore, 2006, pp. 335-340.

[18] S. Wang, Y. Gong, G. Chen, Q. Sun and F. Yang, "Service vulnerability scanning based on service-oriented architecture in Web service environments," *Journal of Systems Architecture*, vol. 59, pp. 731-739, 2013.

[19] E. Y. Song and K. B. Lee, "Service-oriented Sensor Data Interoperability for IEEE 1451 Smart Transducers," in *IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, Singapore, 2009, pp. 1043 - 1048.

[20] "LPC1769/68/67/66/65/64/63 32-bit ARM Cortex-M3 microcontroller; up to 512 kB flash and 64 kB SRAM with Ethernet, USB 2.0 Host/Device/OTG, CAN," NXP, Product data sheet, 2015, pp. 1-90.

[21] J. Bungo, "Embedded systems programming in the cloud: A novel approach for academia," *IEEE Potentials*, vol. 30, no. 1, p. 17–23, 2011.

[22] R. A. v. Engelen and K. A. Gallivan, "The gSOAP toolkit for web services and peer-to peer computing networks," in *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID)*, Berlin, 2002, pp. 128-136.

[23] N. Bežanić and I. Popović, "Virtual transducers in service-oriented smart transducers network," in *21. telekomunikacioni forum (TELFOR)*, Beograd, 2013, pp. 813-816.

[24] N. Bežanić and I. Popović, "Service-oriented Implementation Model for Smart Transducers Network," *Computer Standards & Interfaces*, vol. 38C, pp. 78-83, 2015.

[25] N. Bežanić, I. Radovanović and P. Ivan, "Implementacija servisno orijentisane arhitekture u mreži pametnih pretvarača," in *YU INFO*, Kopaonik, 2012, pp. 544-548.

[26] N. Bežanić and I. Popović, "Service-oriented sensor network for environmental monitoring," in *20. telekomunikacioni forum, TELFOR*, Beograd, 2012, pp. 1544 – 1547.

[27] “KTY81 series, Silicon temperature sensors,” NXP, Product data sheet, 2008, pp. 1-15.

[28] “SHT1x / SHT7x, Humidity & Temperature Sensor,” Humidity & Temperature Sensor, 2007, Product data sheet, pp. 1-10.

[29] L. D. Xu, W. He and S. Li, “Internet of things in industries: A survey,” *IEEE Transactions on Industrial Informatics,* vol. 10, pp. 2233-2243, 2014.

[30] N. Komoda, “Service Oriented Architecture (SOA) in Industrial Systems,” in *IEEE International Conference on Industrial Informatics*, 2006, pp. 1-5.

[31] T. Cucinotta, A. Mancina, G. Anastasi, G. Lipari, L. Mangeruca, R. Checcozzo and F. Rusina, “A Real-Time Service-Oriented Architecture for Industrial Automation,” *IEEE Transactions on Industrial Informatics,* vol. 5, pp. 267-277, 2009.

[32] S. Karnouskos, A. Colombo, F. Jammes, J. Delsing and T. Bangemann, “Towards an architecture for service-oriented process monitoring and control,” in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, 2010, pp. 1385-1391.

[33] A. Colombo, S. Karnouskos, J. Mendes and P. Leitão, “Industrial Agents in the Era of Service-Oriented Architectures and Cloud-Based Industrial Infrastructures,” *Industrial Agents, Emerging Applications of Software Agents in Industry, Elsevier Science,* pp. 67-87, 2015.

[34] T. Strasser and A. Zoitl, “Distributed Real-Time Automation and Control - Reactive Control Layer for Industrial Agents,” *Industrial Agents, Emerging Applications of Software Agents in Industry, Elsevier Science,* pp. 89-107, 2015.

[35] R. S. Alonso, D. I. Tapia, J. Bajo, Ó. García, J. F. d. Paz and J. M. Corchado, “Implementing a hardware-embedded reactive agents platform based on a service-oriented architecture over heterogeneous wireless sensor networks,” *Ad Hoc Networks,* vol. 11, pp. 151-166, 2013.

[36] K. Nagorny, A. W. Colombo and U. Schmidtmann, “A service- and multi-agent-oriented manufacturing automation architecture: An IEC 62264 level 2 compliant implementation,” *Computers in Industry, Special Issue on Sustainable Interoperability: The Future of Internet Based Industrial Enterprises,* vol. 63, pp. 813-823, 2012.

[37] P. Leitao, V. Marik and P. Vrba, “Past, Present, and Future of Industrial Agent Applications,” *IEEE Transactions on Industrial Informatics,* vol. 9, pp. 2360-2372, 2013.

[38] N. Bežanić, I. Popović and A. Rakić, “Integration of Signal Prediction Service in Service Oriented Architecture,” in *12th Symposium on Neural Network Applications in Electrical Engineering (NEUREL)*, Belgrade, 2014, pp. 201-206.

[39] Z. Lixian, G. Huijun and O. Kaynak, “Network-Induced Constraints in Networked Control Systems—A Survey,” *IEEE Transactions on Industrial Informatics,* vol. 9, pp. 403-416, 2013.

[40] A. Doulamis and N. Matsatsinis, “Visual understanding industrial workflows under uncertainty on distributed service oriented architectures,” *Future Generation Computer Systems,* vol. 28, p. 605–617, 2012.

[41] R. A. Gupta and C. Mo-Yuen, “Networked Control System: Overview and

Research Trends," *IEEE Transactions on Industrial Electronics,* vol. 57, pp. 2527-2535, 2010.

[42] J. Baillieul and P. J. Antsaklis, "Control and Communication Challenges in Networked Real-Time Systems," in *Proceedings of the IEEE,* 2007, pp. 9-28.

[43] O. Esquivel-Flores, H. Benítez-Pérez and J. Ortega-Arjona, "Issues on Communication Network Control System Based Upon Scheduling Strategy Using Numerical Simulations," in *D.M. Andriychuk (Ed.) Numerical Simulation - From Theory to Industry,* InTech, 2012, pp. 49-66.

[44] J. Heaton, Programming Neural Networks with Encog 2 in Java, Rev. 1, Heaton Research Inc, 2010, pp. 1-479.

[45] J. A. Rossiter, Model-Based Predictive Control: A Practical Approach (Control Series), CRC Press, 2004.

[46] N. Bežanić, J. Popović-Božović, I. Popović, G. Dimić and V. Milutinović, "Large dataset encryption on the Maxeler platform: a service-oriented approach," in *Tenth International Summer School on Advanced Computer Architecture and Compilation for High-Performance and Embedded Systems, ACACES 2014, Poster Abstracts,* Fiuggi, Italy, 2014, pp. 17-20.

[47] O. Mencer, "Maximum performance computing for exascale applications," in *International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS),* Samos, 2012, pp. 3-3.

[48] V. Milutinovic, "SuperComputers: ControlFlow versus DataFlow," in *2nd Mediterranean Conference on Embedded Computing (MECO)*, 2013, pp. 1-1.

[49] "Multiscale Dataflow Programming," Maxeler, Tutorial, 2012, pp. 1-168.

[50] S. Stojanovic, D. Bojic, M. Bojovic, M. Valero and V. Milutinovic, "An overview of selected hybrid and reconfigurable architectures," in *IEEE International Conference on Industrial Technology (ICIT),* Athens, 2012, pp. 444-449.

[51] S. Baktir and E. Savas, "Highly-Parallel Montgomery Multiplication for Multi-core General-Purpose Microprocessors," in *27th International Symposium on computer and Information Sciences (ISCIS), Computer and Information Sciences III*, Paris, 2012, pp. 467-476.

[52] C. Kaya Koc, T. Acar and B. S. J. Kaliski, "Analyzing and Comparing Montgomery Multiplication Algorithms," in *IEEE Micro,* 1996, pp. 26-33.

[53] N. Bežanić, J. Popović-Božović, V. Milutinović and I. Popović, "Implementation of the RSA Algorithm on a DataFlow Architecture," *Transactions on Internet Research, IPSI BgD,* vol. 9, no. 2, pp. 11-16, 2013.

[54] "Dataflow Programming with MaxCompiler," Maxeler, Tutorial, 2012, pp. .

[55] A Simple Network Management Protocol (SNMP), Available online (December 2015): https://www.ietf.org/rfc/rfc1157.txt, 1990, Request for Comments 1157.

[56] Management Information Base for Network Management of TCP/IP-based internets: MIB-II, Available online (December 2015): https://www.ietf.org/rfc/rfc1213.txt, 1991, Request for Comments 1213.

[57] Structure of Management Information Version 2 (SMIv2), Available online (December 2015): https://tools.ietf.org/html/rfc2578, 1999, Request for Comments: 2578.

[58] C. T. T. K. B. V. Balázs Scherer, “SNMP-based Approach to Scalable Smart Transducer Networks,” in *Instrumentation and Measurement Technology Conference (IMTC)*, 2003, pp. 721 - 725.

[59] S. A. Hussain and D. Gurkan, “Management and Plug and Play of Sensor Networks Using SNMP,” *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 5, pp. 1830 - 1837, 2011.

[60] S. Gumudavelli, D. Gurkan, S. A. Hussain and R. Wang, “A Network Management Approach for Implementing the Smart Sensor Plug and Play,” in *IEEE Sensors Applications Symposium (SAS)*, Limerick, 2010, pp. 261 - 264.

[61] S. A. Hussain, D. Gurkan and S. Gumudavelli, “Design of a Management Information Base (MIB) for a Smart Sensor Network,” in *Instrumentation and Measurement Technology Conference (I2MTC)*, Austin, TX, 2010, pp. 1126 - 1130.

[62] G. M. Waleed and R. B. Ahmad, “Security protection using simple object access protocol (SOAP) messages techniques,” in *International Conference on Electronic Design (ICED)*, Penang, 2008, pp. 1-6.

[63] N. Bežanić, R. Đurić and I. Popović, “Power Management in Service-oriented Smart Transducers Network,” in *22. telekomunikacioni forum (TELFOR)*, Beograd, 2014, pp. 991-994.

[64] S. Liu, Adaptive Power Management for Energy Harvesting Real-time embedded Systems, Doctoral dissertation, State University of New York at Binghamton, 2010.

[65] I. Ratković, N. Bežanić, O. S. Ünsal, A. Cristal and V. Milutinović, “An Overview of Architecture-Level Power- and Energy-Efficient Design Techniques,” *Advances in Computers*, vol. 98, pp. 1-57, 2015.

[66] F. M. Kovatsch, Scalable Web Technology for the Internet of Things, Doctoral dissertation, ETH ZURICH, 2015.

[67] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, “Internet of Things: A Survey on Enabling Technologies, Protocols and Applications,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347 - 2376, 2015.

Biografija autora

Nikola Bežanić je rođen 08.03.1984. godine u Čačku, gde je i završio osnovnu školu i gimnaziju. Elektrotehnički fakultet Univerziteta u Beogradu upisao je 2003. godine. Diplomirao je sa prosečnom ocenom 8,35 na Odseku za elektroniku, odbranom diplomskog rada pod nazivom „Kontrola radne temperature audio uređaja infracrvenim termometrom“, 08.12.2009. godine. Na istom fakultetu upisao je i diplomske akademske studije master, modul Elektronika, koje je završio sa prosečnom ocenom 9,71, odbranom master rada pod nazivom „Implementacija servisno orijentisane arhitekture u mreži pametnih pretvarača“, 26.10.2011. godine.

Doktorske studije na Elektrotehničkom fakultetu Univerziteta u Beogradu, modul Elektronika, upisao je 2012. godine gde je trenutno student treće godine. Od 01.10.2013. godine zaposlen je na Elektrotehničkom fakultetu Univerziteta u Beogradu, gde je angažovan na projektu tehnološkog razvoja Ministarstva prosvete, nauke i tehnološkog razvoja pod nazivom *„Razvoj i modelovanje energetski efikasnih, adaptabilnih, višeprocesorskih i višesenzorskih elektronskih sistema male snage“*, TR-32043.

Nikola Bežanić je započeo naučno-istraživačko iskustvo 2009. godine u saradnji sa *Barcelona Supercomputing Center* (BSC) španskim nacionalnim centrom za superračunare, gde se u okviru projektnih aktivnosti u periodu od 2009-2011 bavio tehnikama uštede energije na nivou arhitekture računara. Dalje iskustvo je stekao na projektu Ministarstva prosvete, nauke i tehnološkog razvoja, radeći na istraživanju servisno orijentisanih senzorskih mreža. Kandidat je koautor dva naučna rada objavljena u časopisima sa ISI liste u kategorijama M22 i M23, jednog rada u časopisu nacionalnog značaja kategorije M52, jedanaest radova objavljenih na međunarodnim i domaćim konferencijama kategorije M33 i M63. Takođe, učestvovao je u izradi pet tehničkih rešenja kategorije M85 i M84.

Прилог 1.

Изјава о ауторству

Потписани Никола Бежанић
Број уписа 5043/2012

Изјављујем

да је докторска дисертација под насловом:

Модел имплементације сервисно оријентисаних мрежа паметних претварача

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио ауторска права и користио интелектуалну својину других лица.

У Београду, 24.12.2015.

Потпис докторанда

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Никола Бежанић
Број уписа 5043/2012
Студијски програм Електроника
Наслов рада Модел имплементације сервисно оријентисаних мрежа паметних претварача
Ментор Доц. др Иван Поповић

Потписани Никола Бежанић

изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду**.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, 24.12.2015.

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Модел имплементације сервисно оријентисаних мрежа паметних претварача

која је моје ауторско дело.

Дисертацију са свим прилозима предао сам у електронском формату погодном за трајно архивирање.
Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио.

1. Ауторство
2. Ауторство – некомерцијално
3. Ауторство – некомерцијално – без прераде
4. Ауторство – некомерцијално – делити под истим условима
5. Ауторство – без прераде
6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на крају).

Потпис докторанда

У Београду, 24.12.2015.