Primena ZeroMQ i Protocol Buffers za komunikaciju između aplikacija u SCADA sistemu

A. Cvetković

***Sadržaj* — Supervisory Control And Data Acquisition (SCADA) je računarski kontrolisan sistem za nadgledanje i upravljanje industrijskim procesima. Sastoji se od različitih komponenata za prikupljanje i prenos podataka i od softvera koji obezbeđuje praćenje i kontrolu postrojenja sa jedne lokacije. Da bi sistem ispravno funkcionisao neophodna je brza i pouzdana komunikacija između svih SCADA aplikacija i kontrolnih uređaja. U radu je ukratko opisana arhitektura SCADA sistema i postojeći problemi sa komunikacijom između SCADA aplikacija. Detaljno je opisano unapređenje već postojećeg mehanizma komunikacije bazirano na RPC (Remote Procedure Call) tehnologiji korišćenjem ZeroMQ i Google Protocol Buffers biblioteka. Novo rešenje znatno smanjuje kompleksnost i obezbeđuje proširivost celog sistema upravljanja.**

***Ključne reči* — SCADA, RPC, ZeroMQ, Protocol Buffers, komunikacija, distribucija događaja.**

I. UVOD

INDUSTRIJSKI kontrolni sistem (**I**ndustrial **C**ontrol **S**ystem - ICS) se koristi u industrijskoj proizvodnji za nadgledanje i kontrolu opreme ili mašina i najčešće se upotrebljava u industrijama kao što su električna, naftna, gasna, za vodovod i kanalizaciju. **S**upervisory **C**ontrol **A**nd **D**ata **A**cquisition – SCADA [1] je tip industrijskog kontrolnog sistema koji predstavlja računarski kontrolisan sistem za nadgledanje i upravljanje industrijskim procesima. SCADA se sastoji od sistema za prikupljanje i prenos podataka, kao i od softvera za prezentaciju koji obezbeđuje centralizovano praćenje i kontrolu postrojenja. Ovi sistemi su dizajnirani za prikupljanje informacija sa

A. Cvetković, Institut Mihajlo Pupin, Volgina 15, 11050 Beograd, Srbija; (e-mail: aleksandar.cvetkovic@pupin.rs).

terena, njihov prenos do centralnog računara i prikaz informacija operateru u grafičkom ili tekstualnom obliku. Pomoću SCADA sistema operater može da prati i upravlja celokupnim postrojenjem u realnom vremenu sa jedne lokacije. Glavna razlika između SCADA i ostalih kontrolnih sistema je ta što je SCADA sistem prvenstveno namenjen za kontrolu velikih, geografski razuđenih sistema.

Komponente SCADA sistema su SCADA server, HMI (Human Machine Interface) aplikacije [2], arhivski server, veb server i programabilni logički kontroleri [3]. Svaka od navedenih komponenta se sastoji od neophodnog hardvera i softvera. Podela sistema na više komponenata, koje su međusobno povezane, obezbeđuje mogućnost različitog konfigurisanja sistema. Centralna komponenta je SCADA server aplikacija koja prikuplja i obrađuje podatke sa udaljenih kontrolnih uređaja i pruža servise svim aplikacijama kojima su ti podaci potrebni. Udaljeni kontrolni uređaji su programabilni logički kontroleri, udaljene jedinice [4] i razni inteligentni merni uređaji [5] sa kojima SCADA server komunicira preko specijalizovanih industrijskih protokola. SCADA aplikacije koriste ove prikupljene podatke o merenjima kako bi generisale izveštaje (aplikacije izveštajnog podsistema) ili kako bi prikazale stanje sistema upravljanja (HMI aplikacije). Ova komunikacija se obavlja preko standardnih servisa koji su široko prihvaćeni za potrebe razmene podataka između aplikacija. Ukratko se može reći da SCADA server obezbeđuje vezu između aplikacija i udaljenih kontrolnih uređaja, što je veoma bitno jer aplikacije i uređaji komuniciraju preko različitih komunikacionih protokola. Server se često izvršava na Linux operativnom sistemu zbog njegove detaljnije konfiguracije i velike pouzdanosti.

SCADA Arhiver (Historian) je servis koji u bazi čuva prikupljene podatke sa različitih kontrolnih uređaja. Arhiviranje podataka iz sistema je veoma bitno za izveštavanje, analiziranje i sprečavanje budućih nesreća kroz analizu generisanih izveštaja. Podaci koji se arhiviraju su vrednosti, statusi procesnih veličina u različitim vremenskim trenucima, događaji, alarmi i akcije operatera. Arhiver obezbeđuje i servise preko kojih druge aplikacije mogu da pristupe arhiviranim podacima i da ih prikažu u arhivskim graficima ili da generišu dnevne, mesečne i godišnje izveštaje.

Podaci iz sistema upravljanja, koji se prikupljaju na SCADA serveru i koji se skladište pomoću arhivskog servera, moraju se prezentovati operateru koji je zadužen za upravljanje sistemom. Na osnovu prezentovanih podataka operater nadgleda i kontroliše proces upravljanja. HMI je aplikacija koja obezbeđuje interakciju između čoveka i mašina i veoma je važna komponenta SCADA sistema. Cilj ove aplikacije je da operateru pojednostavi upravljanje

postrojenjem kroz jednostavan, efikasan i prihvatljiv korisnički interfejs. HMI aplikacija podatke prikuplja sa SCADA i arhivskog servera i krajnjem korisniku omogućuje prikaz željenog skupa informacija, izmenu određenog skupa podataka i izdavanje upravljačkih naloga. Veoma važan zahtev jeste prenosivost HMI aplikacije na sve važnije operativne sisteme kao što su Linux i Windows.

Veb server, kao SCADA komponenta, obezbeđuje udaljeni pristup sistemu upravljanja pomoću internet pregledača i mobilnih uređaja. Koristi se za obradu i prenos sadržaja veb stranica do udaljenih klijenata. Veb stranice se prave dinamički na osnovu podataka prikupljenih sa SCADA i arhivskog servera. One sadrže razne izveštaje, liste događaja, trenutne i arhivske grafike i dinamičke prikaze. Zbog sigurnosti se često zabranjuje komandovanje pomoću veb aplikacije, već se dozvoljava samo nadgledanje i analiza sistema upravljanja.

Kontrolni sistemi su veoma raznovrsni po pitanju programskih jezika, tipova aplikacija i različitih komunikacionih tehnologija koje se koriste za njihovu implementaciju. SCADA server i arhiver su aplikacije čije su karakteristike pouzdanost i efikasnost -- zato se često realizuju u programskim jezicima koji su bliži hardveru (na primer C) i koji dosta zavise od funkcija operativnog sistema. Korišćenje veb servera donosi nove veb tehnologije kao što su HTML5, JavaScript, WebGL, JavaEE i PHP. Današnje HMI aplikacije moraju da budu vizualno lepe i prenosne i zato se za njihovu realizaciju koriste programski jezici višeg nivoa koji imaju ugrađene funkcionalnosti za rad sa grafikom, za rad sa korisničkim interfejsima i za mrežnu komunikaciju (Java, Qt, Python). Takođe, prenosivost aplikacija na Windows i Linux operativne sisteme je veoma bitan zahtev koji se postavlja pred sve komponente SCADA sistema.

Komunikacija između klijent i server aplikacija, kao i između SCADA servera i udaljenih kontrolnih uređaja je još raznovrsnija jer ne postoji jedan standard koji bi specificirao jedinstveni komunikacioni protokol. Zato danas postoji više SCADA rešenja koja su sačinjena od različitih komunikacionih tehnologija i to od veb servisa, RPC (Remote Procedure Call), CORBA, OPC, i još mnogo drugih. Postojeće rešenje za komunikaciju između SCADA aplikacija bazirano je na RPC [6] tehnologiji koja omogućuje poziv funkcija udaljenih aplikacija**.** RPC je veoma efikasan mehanizam koji od programera skriva nepotrebne detalje u vezi uspostavljanja i održavanja konekcije ka udaljenom računaru. Takođe, prednost RPC-a je što se načini poziva funkcija ka udaljenoj i lokalnoj aplikaciji ne razlikuju. Zbog jednostavnijeg korišćenja napravljena je komunikaciona SCADA biblioteka

koja je bazirana na RPC mehanizmu i koja skriva kompleksnost komunikacije sa SCADA serverom. Postoje dve implementacije ove biblioteke za Java i C/C++ programski jezik zato što su ovo dva najkorišćenija jezika za razvoj SCADA aplikacija. Za Java implementaciju korišćena je Remote Tea [7] biblioteka koja se bazira na Sun ONC/RPC specifikaciji. Veliki problem je slaba podrška zajednice za razvoj ovih biblioteka (Remote Tea je poslednji put ažurirana 2008. godine), što se ogleda u njenom nepredvidljivom radu na novijim operativnim sistemima. Još veći problem predstavlja nemogućnost jednostavnog formiranja različitih mrežnih obrazaca koji su neophodni za efikasnu komunikaciju između SCADA aplikacija. Naime, RPC mehanizam je prilagođen za klijent-server način komunikacije, gde klijent aplikacija šalje serveru zahtev, a server obrađuje zahtev i vraća klijent aplikaciji rezultat obrade. Ovakav način komunikacije odgovara u slučaju dohvatanja vrednosti i statusa za merenja sa SCADA servera, dohvatanja arhivskih podataka ili kada se izdaju upravljački nalozi (komanduje). Problem nastaje prilikom distribucije događaja sa SCADA servera ka svim prijavljenim aplikacijama u mreži. Na svaku promenu fizičke veličine, koja se meri, SCADA server generiše događaj koji treba da upozori operatera na izmene u sistemu upravljanja. Ponekad ovih događaja može biti previše (preko 1000 u sekundi) i zato je neophodan efikasan mehanizam za komunikaciju između servera i više aplikacija istovremeno. Da bi se zadovoljili zahtevi za pravilnu distribuciju događaja, bilo je neophodno da se napiše server za distribuciju događaja koji održava konekcije ka svim udaljenim aplikacijama i koji distribuira nove događaje svakoj aplikaciji posebno. Ovo nije najbolje rešenje zato što se dodatno troše resursi sistema (usled više konekcija) i zato što se mora posebno paziti na ispravnost svake konekcije. Zbog svega navedenog, biblioteka za komunikaciju je postala teška za održavanje i dalje unapređenje. Iako RPC komunikacija poseduje svoje prednosti, ona danas ima više mana zbog kojih se mora tražiti bolji način komunikacije između SCADA aplikacija.

Brza i pouzdana komunikacija između SCADA aplikacija je neophodna za ispravno funkcionisanje sistema upravljanja. Zato novi mehanizam komunikacije mora da zadovolji sledeće zahteve:

- Mala veličina biblioteke, efikasna komunikacija i malo zauzeće resursa računara.
- Mogućnost formiranja različitih obrazaca za razmenu poruka (najvažniji su request/reply i publish/subscribe).
- Sinhrona i asihrona komunikacija.
- Detaljna podešavanja parametara i kontrola komunikacije

(kontrola redova sa porukama, vreme odziva, rad sa greškama, itd).
- Implementacija biblioteka za C/C++, Java i PHP jezike.
- Platformska nezavisnost (rad na Windows, Linux i Android operativnim sistemima).
- Jednostavno učenje i primena.

U ovom radu opisaćemo kako se može unaprediti već postojeći mehanizam komunikacije između SCADA aplikacija i smanjićemo kompleksnost celog sistema upravljanja korišćenjem ZeroMQ i Google Protocol Buffer biblioteka.

II. UPOTREBA ZEROMQ I PROTOCOL BUFFERS BIBLIOTEKA ZA KOMUNIKACIJU IZMEĐU SCADA APLIKACIJA

ZeroMQ [8] je biblioteka za rad sa porukama i koristi se prilikom razvoja distribuiranih i konkurentnih aplikacija. Evropsko veće za nuklearna istraživanja (CERN) je 2011. godine istraživalo načine da se objedine sve aplikacije koje se koriste za upravljanje i analizu rada CERN akceleratora. Studija je imala za cilj da poredi implementacije za protokole otvorenog koda CORBA [9], Ice [10], Thrift [11], ZeroMQ, Qpid [12], i zaključak je bio da ZeroMQ biblioteka najviše odgovara zbog svoje fleksibilnosti i lake prilagodljivosti različitim operativnim sistemima (naročito Linux-u). ZeroMQ koristi redove prilikom rada sa porukama, ali ne koristi broker komponentu. Broker komponenta je nezavisna komponenta koja služi za skladištenje i distribuiranje poruka do udaljenih aplikacija. Problem sa brokerom je što često zauzima previše računarskih resursa i, u slučaju prestanka njegovog rada, raspada se celokupan mehanizam za komunikaciju. Zato se pokreće više brokera koji rade paralelno čime se obezbeđuje veća pouzdanost. Tada se pojavljuje problem sa konfiguracijom i održavanjem rada broker komponenata, što nije jednostavan posao i zahteva konstantno angažovanje administratora sistema. U SCADA sistemima se izbegava upotreba mehanizma komunikacije koji poseduje broker komponentu zbog toga što je distribucija poruka sporija i zahteva više računarskih resursa. ZeroMQ je jednostavnija biblioteka za komunikaciju koja intezivno koristi redove za skladištenje i distribuciju poruka. Ovi redovi su mali, veoma efikasni i nalaze na svakoj strani komunikacije. Nepostojanje centralne "broker" komponente je osobina koja se ceni prilikom razvoja vremenski kritičnih aplikacija jer znatno povećava odziv i pouzdanost celog sistema. Razlozi zbog kojih smo izabrali ZeroMQ biblioteku za primenu u SCADA

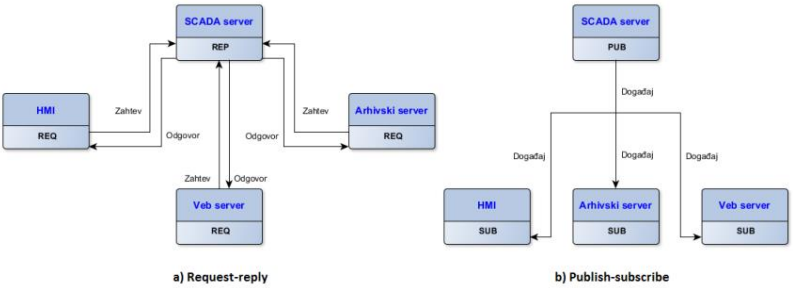
sistemima su:

- Brz i efikasan prenos poruka između SCADA aplikacija.
- Platformska i programska nezavisnost. Postoje implementacije za skoro sve bitnije programske jezike (C/C++, Python, Java, PHP) i za sve bitnije operativne sisteme (Windows, Linux, Android).
- Slanje i primanje poruka se može obavljati sinhrono i asihrono u pozadinskoj niti.
- Automatsko ponovno povezivanje obezbeđuje da se softverske komponente priključuju i odjavljuju dinamički. Ova osobina je bitna jer zbog nje nije bitan redosled startovanja aplikacija i obezbeđuje robusnost jer se servisi mogu ponovo pokretati nakon njihovog pada. SCADA server se može restartovati u slučaju nepravilnog rada i da automatski nastavi distribuciju poruka ka udaljenim SCADA aplikacijama.
- Zbog dobre optimizacije koja minimizuje čuvanje poruka u redovima za poruke.
- Zbog mogućnosti detaljnog podešavanja različitih tipova slanja poruka. Na primer, može se konfigurisati tako da se blokira aplikacija ili nit koja šalje poruku, da se poruka odbaci itd, kada se red prepuni porukama.
- Aplikacije mogu da komuniciraju preko različitih transpornih puteva kao što su in-process, inter-process (IPC), TCP ili multicast (PGM). In-process služi za komunikaciju između niti u procesu, IPC služi za komunikaciju između različitih procesa, Transmission control protocol (TCP) je mrežni protokol za komunikaciju između aplikacija koje se izvršavaju na istom ili različitim računarima i Pragmatic General Multicast (PGM) je mrežni protokol koji obezbeđuje sigurno slanje poruka ka više aplikacija. Poruke se šalju na isti način nezavisno od tipa transportnog puta, što je veoma dobra osobina ZeroMQ biblioteke.
- Moguće je napraviti različite mrežne obrasce (request-reply, publish-subscribe) kako bi se bolje odgovorilo na zahteve sistema upravljanja.
- Poruke zadržavaju svoju veličinu prilikom slanja. Ne postoji podrazumevani format poruke i ona može biti veličine od nekoliko bajtova do gigabajta. Ova osobina nam štedi mrežni protok jer garantuje da se poruke neće proširiti sa dodatnim informacijama neophodnim samo za slanje. U SCADA sistemima

poruke mogu biti različitih veličina.

- Pažljiva obrada grešaka na mreži. Postoji mogućnost ponavljanja neuspele operacije ili prijava greške.
- Zbog svoje efikasnosti ZeroMQ bibiloteka se može koristiti za komunikaciju sa uređajima koji imaju ograničenu memoriju i procesor. Ovakvi namenski uređaji su česti u sistemima upravljanja.

Mrežni soket predstavlja standardizovani način mrežne komunikacije između aplikacija. Aplikacije uglavnom koriste sokete pomoću programskog interfejsa koji obezbeđuje operativni sistem. Kako bi se olakšao posao programerima, ZeroMQ biblioteka koristi i znatno proširuje već postojeće koncepte soketa i poruka. Postoji više tipova ZeroMQ soketa i, u zavisnosti od načina njihovog povezivanja, moguće je napraviti različite mrežne obrasce. Ovi obrasci omogućuju pronalaženje najboljeg načina razmene poruka između aplikacija. Poznatiji mrežni obrazci koji se primenjuju za komunikciju sa SCADA aplikacijama prikazani su na slici 1.



Sl. 1. Mrežni obrasci napravljeni pomoću ZeroMQ biblioteke za dohvatanje podataka i distribuciju događaja sa SCADA servera.

SCADA server opslužuje klijent aplikacije podacima koje prikuplja i obrađuje iz procesa kojim se upravlja. Najčešći zahtevi SCADA aplikacija su dohvatanje najnovijih vrednosti i statusa za merenja, pristup arhiviranim podacima, distribucija događaja i izdavanje komandnih naloga. Dohvatanje vrednosti i statusa za merenja, kao i komandovanje se izvršava tako što klijent šalje zahtev serveru i čeka na njegov odgovor. Ovakav tip komunikacije smo realizovali pomoću request-reply obrasca (slika 1 pod a)) koji povezuje više SCADA aplikacija sa serverom. Request-reply [13] je jednostavan i efikasan obrazac koji obezbeđuje dvosmernu komunikaciju i

veoma je čest u klijent-server arhitekturama. Kod izdavanja komandnih naloga bitno je da se obrade sve mrežne greške i da se operater obavesti o uspešnosti operacije, kao i da se definiše maksimalno vreme čekanja za koje se operacija mora izvršiti. Request-reply obrazac se pravi uparivanjem REQ i REP tipova ZeroMQ soketa.

Svaka promena u procesu kojim se upravlja može rezultovati generisanjem događaja koji sadrži detaljan opis šta se zapravo dogodilo. SCADA server generiše i distribuira događaje do prijavljenih HMI aplikacija, koji se zatim prikazuju operateru u listi događaja. Današnji sistemi upravljanja moraju da obezbede jednostavno prijavljivanje aplikacija za komunikaciju sa SCADA serverom i za distribuciju događaja. Ova "otvorenost" sistema je veoma bitna osobina zbog pojave poslovnih aplikacija koje koriste podatke iz procesa za analizu i generisanje izveštaja. Svaka distribucija će imati zahtev da se jednostavno poveže sa SCADA serverom i da dohvati podatke od interesa. Takođe, SCADA server može da izgeneriše i do nekoliko hiljada poruka u sekundi i zato je neopohodan efikasan mehanizam za razmenu poruka preko mreže. Za potrebe distribucije događaja koristili smo publish-subscribe [13] obrazac kod koga server oglašava (prosleđuje) poruke ka svim prijavljenim SCADA aplikacijama (slika 1 pod b)). Klijent aplikacije se mogu dinamički prijavljivati i odjavljivati serveru, čime se postiže veća skalabilnost. Publish-subscribe obrazac se pravi uparivanjem PUB i SUB tipova ZeroMQ soketa, gde na jedan PUB može biti nakačeno više SUB soketa. Publish-suscribe obrazac je posebno pogodan za upotrebu u SCADA sistemima zbog osobine filtriranja poruka koja obezbeđuje da prijavljene aplikacije prime samo one događaje od interesa. Naime, kada se aplikacija prijavi serveru, ona može da zahteva prijem samo određenih događaja vezanih za merenje, grupu merenja, za sva merenja koja pripadaju objektu, polju ili stanici iz procesa kojim se upravlja. Prednost ovog obrasca za slanje poruka jeste što su jasno razdvojene server i klijent aplikacije koje međusobno komuniciraju samo preko poruka. Takođe, velika prednost je proširivost sistema zato što možemo registrovati veliki broj klijent aplikacija bez dodatnog konfigurisanja ili programiranja. ZeroMQ omogućuje da se klijent aplikacija (SUB soket) poveže na više servera za distribuciju poruka (PUB soketa) i tada se pristigli događaji od različitih servera ravnomerno skladište na klijent strani. Ako ne postoje prijavljene SCADA aplikacije, tada će server odbaciti sve nove događaje.

Pored navedenih tipova postoje i PUSH, PULL, DEALER, ROUTER i PAIR tipovi ZeroMQ soketa čijim povezivanjem je moguće dobiti obrasce (mrežne topologije) koji najbolje odgovaraju sistemu upravljanja. Na primer,

korišćenjem PUSH i PULL tipova soketa moguće je napraviti „Parallel Pipeline" obrazac [13] koji se koristi kada se želi paralelna obrada više zadataka. Ovo je česta pojava u super računarstvu gde se veći zadatak deli na više manjih delova i šalje uređajima (često grafičkim karticama) koje paralelno izvršavaju zadatke, a na kraju se rezultati sakupljaju i obrađuju na jednom mestu. PUSH soket služi da distribuira poruke (zadatke i rezultate), dok PULL soket prihavata poruke i obrađuje ih. U SCADA sistemima ovakav obrazac se može iskoristiti prilikom realizacije algoritma za proračun energizovanosti vodova. Ovaj algoritam prolazi kroz celu električnu mrežu koja je predstavljena kao razgranato stablo i na osnovu stanja rasklopne opreme određuje da li su vodovi pod naponom, uzemljeni, itd.

ZeroMQ poruke ne nameću bilo kakav format, već je njihov sadržaj predstavljen kao niz bajtova (blob) proizvoljne veličine. Da bi poruka bila smislena, moramo definisati njen sadržaj kao i način njenog čitanja i pisanja. Za pisanje i čitanje sadržaja poruke koristićemo proizvod Google protocol buffer [14] koji predstavlja platformski i programski nezavisan mehanizam za serijalizaciju strukturiranih podataka. On obezbeđuje serijalizaciju podataka na veoma fleksibilan, efikasan i automatizovan način. Struktura podatka se definiše pomoću posebno dizajniranog jezika za opis interfejsa, a zatim se pomoću kompajlera (programa *protoc*) generišu klase za različite programske jezike koje služe za upis i čitanje podatka. Informacije koje se žele serijalizovati definišu se kao poruke (*messages*) i čuvaju se unutar *.proto* fajla. Serijalizacijom se podaci prevode u binarni format što je dobar izbor zbog boljih performansi, ali lošiji izbor zbog nečitljivosti poruka. Google je razvio Protocol Buffer -- proizvod za svoju upotrebu i napravio je kompajlere protokola za C++, Java i Python programske jezike. Cilj firme Google je bio razvoj jednostavnog i efikasnog načina za čuvanje i razmenu bilo kakve strukture podataka. Danas u Googlu postoji preko 49000 različitih tipova poruka definisanih u više od 13000 *.proto* fajlova.

Protocol Buffer je dizajniran da bude 3 do 10 puta manji i 20 do 100 puta brži od XML formata kojim se uglavnom serijalizuju podaci. Mala veličina poruke je veoma važna kako bi obezbedili što manji protok podatka preko mreže. Zato smo iskoristili Protocol Buffer za serijalizuju događaja, kao i za serijalizaciju zahteva i odgovora od SCADA servera. Pored načina serijalizacije poruka (Google protocol buffer) i načina prenosa poruka (ZeroMQ), neophodno je opisati i format poruka. Veoma je bitno da se izbegnu namenski razvijeni modeli podataka koji su česti u manjim firmama zato što se takva rešenja teško integrišu sa drugim aplikacijama. Ovde ćemo opisati delimičnu implementaciju standarda IEC 61970 [15] koji definiše

format za razmenu podataka i programski interfejs za aplikacije koje se koriste u elektroenergetskim sistemima. Izbor ovakvog formata poruka za upotrebu u SCADA sistemu obezbeđuje nam laku integraciju aplikacija različitih proizvođača, razmenu podataka između distribuiranih sistema i proširivost formata kako bi se obezbedila komunikacija sa novijim sistemima. Po standardu informacije se razmenjuju slanjem poruka koje mogu biti tipa alarmi/događaji, podaci i arhivirani podaci. Poruke sa podacima sadrže informacije o merenjima sa SCADA servera. Podaci se obično odnose na vrednosti i statuse merenja. Poruke tipa alarmi/događaji opisuju događaje koje generiše SCADA server. Mogu da prenose podatke o jednom ili više događaja. Podaci se uglavnom odnose na razlog i kategoriju događaja, izvor koji je izazvao događaj, vreme nastanka i ozbiljnost događaja. Poruke sa arhiviranim podacima sadrže podatke iz arhive u vezi jednog ili više merenja za određeni vremenski interval.

Radi boljeg opisa rada Protocol Buffer mehanizma prikazaćemo definiciju SimpleEvent [16] događaja i AlarmsAndEventsMessage [16] objekta koji se sastoji od više događaja. SimpleEvent je tip poruke koji sadrži najosnovnije podatke koji opisuju događaj, i to:

TABELA 1: OPIS SIMPLEEVENT DOGAĐAJA.

Naziv	***Broj***	***Tip***	***Opis***
message	1	String	Tekst sa detaljnim opisom događaja.
reason	1	String	Razlog nastanka događaja.
severity	0..1	Integer	Ozbiljnost događaja. Ovo je broj koji pripada opsegu 1-1000.
source	0..1	String	Izvor koji je proizveo događaj. Može biti bilo koji resurs (program, merenje ili oprema).
time_stamp	1	Date	Vreme kada se događaj dogodio.
type	0..1	Enum	Tip događaja koji može biti ALARM, INFORMATION, FAILURE, itd.
property_values	0..*	PropertyValue	Opcioni niz *PropertyValue* objekata koji mogu da pripadaju događaju.

			PropertyValue je objekat koji povezuje naziv i vrednost neke osobine. Vrednost može biti ceo broj, decimalni broj, string ili logička vrednost.

TABELA 2: DEFINICIJA SIMPLEEVENT I ALARMSANDEVENTSMESSAGE PORUKA POMOĆU PROTOCOL BUFFERS.

SimpleEvent	***AlarmsAndEventsMessage***

message SimpleEvent {
required string message = 1;
required string reason = 2;
optional int32 severity = 3;
optional string source = 4;
required string time_stamp = 5;
enum EventType{
ALARM=1;
INFORMATION=2;
FAILURE=3;
}
optional EventType type=6 [default=ALARM];
 	
message PropertyValue {
required string name = 1;
enum PropertyType {
INTEGER=1;
DOUBLE=2;
STRING=3;
BOOL=4;
}
required PropertyType type=2;
optional int32 integer_value=3;
optional double double_value=4;
optional string string_value=5;
optional bool bool _value=6;
}
repeated PropertyValue property_values = 7;
}
	message AlarmsAndEventsMessage {
	repeated SimpleEvent events=1;
	}

Poruka u Protocol Buffers-u se sastoji od više tipiziranih polja. Od osnovnih tipova polja podržani su bool, int32, float, double i string. Moguće je da se kao polja koriste i drugi tipovi poruka kao što je slučaj sa PropertyValue tipom u SimpleEvent poruci. U slučaju da želimo da polje poseduje predefinisane vrednosti, možemo da definišemo enum tip (primer su EventType i PropertyType u poruci). Dakle pomoću Protocol Buffers možemo da definišemo složene poruke za komunikaciju između SCADA aplikacija. Polje je označeno jedinstvenim brojem koji se koristi prilikom serijalizacije i deserijalizacije poruka. Takođe, svako polje može biti označeno kao obavezno (*required*), opciono (*optional*) i da se ponavlja (*repeated*). Google protocol buffers zahteva da obavezna polja imaju vrednost, dok opciono polje može, ali ne mora da poseduje vrednost. U opciono polje možemo da postavimo predefinisanu vrednost (pogledati polje *type* u SimpleEvent poruci). Polja koja se ponavljanju su zapravo dinamički nizovi koji mogu da sadrže nula ili više elemenata.

Poruke generisane pomoću Protocol Buffers-a su pogodne za upotrebu u SCADA sistemima zbog mogućnosti naknadnog menjanja njihove definicije. Često se dešava da se vremenom SCADA poruke proširuju dodatnim informacijama i zato je neophodno da obezbedimo kompatibilnost starih aplikacija sa novijim tipovima poruka. Ovo je moguće ako ne menjamo jedinstvene brojeve i ako ne brišemo ili ne dodajemo obavezna polja u poruci. Moguće je brisati i dodavati opciona ili polja koja se ponavljaju, ali samo sa jedinstvenim brojevima koji se nikada ranije nisu koristila u poruci. Na ovaj način će stare aplikacije čitati nove poruke i ignorisati nova polja. Ako je neko opciono polje izbrisano, tada će stara aplikacija da pročita njegovu predefinisanu ili praznu vrednost. Nove aplikacije će moći da čitaju i poruke u starom formatu. Ova osobina je idealna zato što nam obezbeđuje jednostavnije održavanje i proširenje svih SCADA komponenata.

III. TEST PERFORMANSI

Testiranje komunikacije pomoću ZeroMQ i Protocol Buffer biblioteka izvršićemo na server mašini (Intel Core i7-4770K, 3.5GHz, 16GB RAM) i na klijent mašinama (Intel Core 2 Quad Q8200, 2.33GHz, 4GB RAM). Komunikacija treba da se pouzdano i efikasno izvršava na GbE mrežama, ali i na sporijim zato što su sporije mreže veoma raspostranjene u starijim sistemima upravljanja. Test ćemo smatrati uspešnim ako se zadovolje sledeći zahtevi:

- *Test 1*: razmena 3000 poruka veličine 30B u sekundi preko request-reply mrežnog obrasca. Ovo se koristi prilikom

dohvatanja podataka za merenje (vrednosti i statusa) sa SCADA servera.

- *Test 2*: razmena 10 većih poruka (veličine 210KB) u sekundi preko request-reply mrežnog obrasca. Ovo se koristi prilikom dohvatanja podataka za grupu merenja (vrednosti i statusa) sa SCADA servera.
- *Test 3*: distribucija 5000 SimpleEvent događaja do deset klijent aplikacija za manje od jedne sekunde korišćenjem publish-subscribe mrežnog obrasca. Ovo je standarni zahtev koji se postavlja za svaki SCADA sistem i koristi se za distribuciju novih događaja vezanih za pojedinačna merenja.
- *Test 4*: distribucija 10 AlarmsAndEventsMessage poruka sa po 1000 SimpleEvent događaja do deset klijent aplikacija za manje od jedne sekunde korišćenjem publish-subscribe mrežnog obrasca. Koristi se prilikom dohvatanja poslednjih 1000 događaja sa SCADA servera nakon prijave HMI aplikacije.
- Uspešan rad stare aplikacije sa novim (izmenjenim) formatom poruke i uspešan rad nove aplikacije sa starijim formatom poruke.
- Uspešno automatsko ponovno povezivanje usled gašenja i ponovnog pokretanja server i klijent aplikacije. Već pokrenute aplikacije treba da nastave normalan rad nakon ponovnog pokretanja oborenih aplikacija.

TABELA 3: BROJ PORUKA U IZVRŠENIM TESTOVIMA.

Test	***Broj poruka u sekundi [msg/s]***
Test 1	4900
Test 2	28
Test 3	43000
Test 4	53

Prilikom testa se merilo vreme potrebno da aplikacija serijalizuje/deserijalizuje podatke i da ih pošalje/primi do druge aplikacije. Pomoću Protocol Buffer-a smo veoma efikasno serijalizovali i deserijalizovali informacije o događajima i ostalim SCADA merenjima. SimpleEvent događaj i SCADA podatak o merenju su, posle serijalizacije, zauzimali oko 150B i 30B, dok je grupa od 1000 događaja (AlarmsAndEventsMessage) zauzimala oko 180KB. Ovo nije uvek slučaj i dosta zavisi od sadržaja događaja, posebno od njegovog opisa koji može biti

različite dužine. U prvom testu smo testirali razmenu poruka (o SCADA merenju) između aplikacije i SCADA servera i uspeli smo da postignemo rezultat od 4900 poruka u sekundi. Drugi test je sličan prvom samo što su poruke koje razmenjujemo mnogo veće. SCADA server je poslao 28 poruke u sekundi sa informacijama o 10000 merenja ukupne veličine 210KB ka udaljenoj aplikaciji. Treći test je distribucija događaja SimpleEvent do deset prijavljenih aplikacija i zabeležen je protok od 43000 poruka u sekundi. Četvrti test je sličan trećem samo što SCADA server šalje veće AlarmsAndEventsMessage poruke koje sadrže 1000 događaja do deset prijavljenih aplikacija. Prva dva testa nam služe za testiranje komunikacije pomoću request-reply mrežnog obrasca, dok druga dva testa testiraju publish-subscribe mrežni obrazac. Rezultati sva četiri testa su veoma zadovoljavajuća.

Testirali smo i kompatibilnost starih i novih aplikacija sa novim i starim formatom poruke. Format poruke smo menjali tako što smo brisali ili dodavali nova opciona polja. Stare aplikacije su uspešno čitale novi format poruke odbacivajući sva nova polja. Takođe, nove aplikacije su uspešno koristile stariji format poruke.

Test automatskog ponovnog povezivanja se sastojao od obaranja i ponovnog pokretanja servera koji je preko publish-subscribe obrazca distribuirao događaje do prijavljenih aplikacija. Po ponovnom pokretanju server je nastavljao da šalje nove događaje ka aplikacijama koje su ranije pokrenute. U aplikacijama nije bilo registrovano da je server bio oboren. Ovo je odlična funkcionalnost ZeroMQ biblioteke koja u pozadini vodi računa o održavanju konekcija. Takođe, prilikom komunikacije preko request-reply mrežnog obrasca, klijent aplikacija se uspešno oporavljala usled obaranja server aplikacije. Pri ponovnom startovanju servera, aplikacija je uspela da se automatski ponovo poveže i da nastavi komunikaciju.

IV. ZAKLJUČAK

ZeroMQ i ProtocolBuffer biblioteke zadovoljavaju skoro sve zahteve za primenu u komunikaciji između SCADA aplikacija. Obe biblioteke se mogu koristiti na skoro svim poznatijim programskim jezicima i operativnim sistemima. Male su veličine i obezbeđuju efikasnu komunikaciju i serijalizaciju/deserijalizaciju podataka sa veoma malim zauzećem računarskih resursa. ZeroMQ obezbeđuje sinhronu i asihronu komunikaciju, kao i mogućnost formiranja različitih obrazaca za razmenu poruka. Raznovrsnost obrazaca nam omogućuje da nađemo najbolje rešenje za

distribuciju događaja i za dohvatanje podataka o merenjima sa SCADA servera.

Rezultati testiranja pokazuju da je razmena poruka pomoću ZeroMQ biblioteke veoma brza i pouzdana. ProtocolBuffer omogućuje menjanje formata poruke, čime je obezbeđeno korišćenje novog formata u starim aplikacijama. Ovo je bitna osobina jer nam omogućuje jednostavnije proširenje i nadogradnju SCADA sistema. ZeroMQ biblioteka poseduje mogućnosti automatskog ponovnog povezivanja u slučaju prekida komunikacije, što je korisna osobina u slučaju distribucije događaja sa SCADA servera do više udaljenih aplikacija. Zbog svoje efikasnosti ZeroMQ bibiloteka se može koristiti za komunikaciju sa uređajima koji imaju ograničenu memoriju i procesor. Ovakvi namenski uređaji su česti u sistemima upravljanja. Na osnovu svega navedenog možemo da zaključimo da su ZeroMQ i Protocol Buffer biblioteke pogodne za komunikaciju između aplikacija u SCADA sistemima.

LITERATURA

[1] S. Boyer. *SCADA: Supervisory Control and Data Acquisition*. ISA-The Instrmentation,Systems, and Automation Society, 2009.

[2] B. Hollifield. *The High Performance HMI Handbook*. Houston, PAS, 2008.

[3] D. Petruzella. *Uvod u programabilne logičke kontrolere*. McGraw-Hill, 2011.

[4] F. Idachaba. *Review of Remote Terminal Unit (RTU) and Gateways for Digital Oilfield delployments.* IJACSA, Vol.3, No. 8, 2012.

[5] Ching-Lai Hor, Peter Crossley. *Knowledge Extraction from Intelligent Electronic Devices.* Springer Link, Lecture Notes in Computer Science Vol. 3400, 2005.

[6] D. Marshall. (1999, may). *Remote Procedure Calls (RPC)* [web page]. Available: http://www.cs.cf.ac.uk/Dave/C/node33.html.

[7] *Remote Tea* [web page]. Available: http://sourceforge.net/projects/remotetea/.

[8] P. Hintjens. *ZeroMQ Messaging for Many Applications*. OReilly, 2013.

[9] Douglas Schmidt. (2012, december). *Distributed Object Computing with CORBA Middleware* [web page]. Available: http:// www.dre.vanderbilt.edu/~schmidt/corba.html.

[10] ZeroC (2013). *Ice 3.5.1 Documentation* [web page]. Available: http://download.zeroc.com/Ice/3.5/Ice-3.5.1.pdf.

[11] Mark Slee, Aditya Agarwal, Marc Kwiatkowski. *Thrift: Scalable Cross-Language Services Implementation.* Paolo Alto, Facebook, 2009.

[12] Apache Qpid. (2013). *Apache Qpid Overview* [web page]. Available: https://qpid.apache.org/overview.html.

[13] Gregor Hohpe, Bobby Woolf. *Enterprise Integration Patterns*. Addison-Wesley, 2004.

[14] Google. (2012). *Protocol Buffers Developers guide* [web page]. Available: https://developers.google.com/protocol-buffers/docs/overview.

[15] *IEC 61970:Energy Management Application Program Interface*. IEC, C.I. 2003.

[16] *IEC 61970:Energy Management Application Program Interface,Part 451:CIS Information ExcangeModel For SCADA.* IEC, C.I. 2003.

ABSTRACT

Supervisory Control And Data Acquisition (SCADA) is a computer-controlled system for monitoring and control of industrial processes. It consists of various components for the collection and transmission of data and the software that provides monitoring and control from one location. Fast and reliable communication between all the SCADA application and control devices is necessary for reliable operation of the system. This paper briefly describes architecture of SCADA system and the existing problems with communication between SCADA applications. Also, this paper describes improvement of the existing communication mechanism based on RPC (Remote Procedure Call) technology using ZeroMQ and Google Protocol Buffers library. The new solution greatly reduces the complexity and ensures scalability of the entire SCADA system.

Communication between SCADA applications using ZeroMQ and Protocol Buffers

A. Cvetković