Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.7
DOI: https://doi.org/10.24867/30BE38Stjepanovic

JEDNO REŠENJE SIMULACIJE SOME/IP KOMUNIKACIJE NA ELEKTRONSKOJ KONTROLNOJ JEDINICI (ECU)

ONE SOLUTION FOR SOME/IP COMMUNICATION SIMULATION ON ELECTRONIC CONTROL UNIT (ECU)

Jelena Stjepanović, *Fakultet tehničkih nauka, Novi Sad*

Oblast – OBRADA SIGNALA

Kratak sadržaj – *Ovaj rad se bavi kreiranjem servera i klijenta u okviru jedne elektronske kontrolne jedinice (Electronic Contol Unit - ECU) i uspostavom SOME/IP (Scalable Service-Oriented MiddlewarE over IP) komunikacije između istih. U uvodu je rečeno nešto više o elektronskoj kontrolnoj jedinici i samoj temi rada. U drugom poglavlju opisana je standardizovana softverska arhitektura u automobilskoj industriji, dok je celo treće poglavlje posvećeno SOME/IP komunikaciji. U četvrtom poglavlju se nudi koncept rešenja, dok peto i šesto poglavlje opisuju njegovu realizaciju i verifikaciju – respektivno. U sedmom poglavlju izveden je zaključak, a u osmom je dat pregled literature.*

Ključne reči: *ECU, SOME/IP, komunikacija, server, klijent, protokol*

Abstract – *This paper deals with the creation of a server and a client within one electronic control unit (Electronic Control Unit - ECU) and the establishment of SOME/IP (Scalable Service-Oriented Middleware over IP) communications between them. In the introduction, something more was said about the electronic control unit and the topic of the work itself. The second chapter describes the standardized software architecture in the automotive industry, while the entire third chapter is devoted to SOME/IP communication. The fourth chapter offers the concept of the solution, while the fifth and sixth chapters describe its implementation and verification - respectively. In the seventh chapter, a conclusion is drawn, and in the eighth, an overview of the literature is given.*

Keywords: *ECU, SOME/IP, communication, server, client, protocol*

1. UVOD

Elektronska kontrolna jedinica (*electronic control unit* – ECU), kao jedan od najvažnijih delova automobila, predstavlja računar koji izvršava sve operacije vezane za pravilan rad i funkcionisanje vozila.
U vozilu postoji više jedinica koje imaju različite funkcije. Samostalna elektronska kontrolna jedinica, koja nije povezana sa ostalim jedinicama nema veliki značaj niti može ostvariti bilo kakvu funkciju. Pravi smisao ECU se postiže tek uspostavom komunikacije, odnosno uvezivanjem svih ECU datog vozila u komunikacionu mrežu gde će promene svake od njih biti redovno i pravovremeno ažurirane, a njihove funkcionalnosti ostvarene kroz međusobno primanje ili slanje poruka i odgovarajuće ponašanje u skladu sa istim.

Kada je u pitanju komunikacija, u automobilskoj industriji se sve više potencira SOME/IP (*Scalable Service-Oriented MiddlewarE over IP*) protokol kao najadekvatniji i po mnogim merilima najfleksibilniji. S obzirom na to da popularnost ovog protokola sve više raste, interesantno je ispitati i implementirati neke od njegovih funkcionalnosti.

Ovaj rad se bavi kreiranjem servera i klijenta u okviru jedne ECU i uspostavom SOME/IP komunikacije između istih. Inovativnost koju bi donela implementacija navedenog ogleda se u činjenici da se SOME/IP protokol uglavnom koristi za komunikaciju između dve različite ECU. S obzirom na to da se u okviru date ECU uglavnom nalazi više softverskih komponenti, korisno je znati da se komunikacija između datih komponenti može ostvariti uz sve mogućnosti SOME/IP komunikacije, pa u tom smislu nije potrebno izmeštati softverske komponente na različite ECU, niti pristajati na kompromis u smislu korišćenja nekog vida interne komunikacije.

2. Softver za automobile – AUTOSAR

AUTOSAR (*AUTomotive Open System ARchitecture*), predstavlja otvorenu, standardizovanu softversku arhitekturu za automobilsku industriju. AUTOSAR je standardizovao dve softverske platforme – *Classic* i *Adaptive*. Danas je AUTOSAR *Classic* u širokoj upotrebi i takođe je prvi izbor za namenske elektronske kontrolne jedinice sa visokim standardima u pogledu sigurnosti i determinističkog izvršenja. AUTOSAR *Classic Platform* arhitektura se raspoznaje na najvišem nivou apstrakcije između tri softverska sloja koja se pokreću na mikrokontroleru: aplikacija, izvršno okruženje i osnovni softver [1].

3. SOME/IP

SOME/IP je razvijen od strane BMW grupe 2011. godine. Kao što ime govori, reč je o rešenju u vidu komunikacionog srednjeg sloja koje se oslanja na TCP/IP (*Transmission Control Protocol*) stek. Dizajniran je da se prilagodi uređajima različitih veličina i različitim

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio prof. dr Dejan Vukobratović.

operativnim sistemima, takođe, podržava funkcije *Infotainment* domena kao i druge domene u vozilu, što mu omogućava da se koristi kao zamena za MOST (*Media Oriented Systems Transport*), kao i zamena za tradicionalna CAN (*Controller Area Network*) scenarija.

3.1. Ključne funkcionalnosti SOME/IP protokola

Neke od glavnih funkcionalnosti SOME/IP protokola su [2]:

• Obezbeđuje servisno orijentisanu komunikaciju unutar mreže

• Podržava širok opseg funkcionalnosti kao posredni softver: *Remote Procedure Call* (RPC), *Service Discovery* (SD), *Publish/Subscribe (Pub/Sub)*

• Segmentacija UDP (*User Datagram Protocol*) poruke (omogućavajući transport velikih SOME/IP poruka preko UDP-a bez potrebe za fragmentacijom)

• Može biti implementiran na različitim operativnim sistemima ili čak na ugrađenim sistemima koji nemaju operativni sistem

• Koristi se za klijent/server serijalizaciju unutar ECU

3.1.1 SOME/IP kao vrsta RPC-a

RPC protokol olakšava komunikaciju između umreženih računara. Program na jednom računaru na mreži koristi RPC da bi došao do programa na drugom računaru na mreži bez poznavanja detalja mreže.

Servis koji nudi SOME/IP može biti sačinjen od nula (dozvoljen je prazan servis) ili više metoda (*methods*), događaja (*events*), ili polja (*fields*) [3].

• Metode donose mogućnost klijentu da izdaje RPC koji se izvršava na strani servera. Razlikuju se *Request/Response* metode i *Fire&Forget* metode.

Request/Response predstavlja jedan od najčešćih obrazaca komunikacije. Jedan učesnik komunikacije (klijent) šalje poruku zahteva na koju odgovara drugi učesnik komunikacije (server). *Fire&Forget* predstavlja oblik komunikacije zasnovan na upućenom zahtevu serveru od strane klijenta bez povratne poruke, odnosno odgovora od strane servera.

• Događaji predstavljaju podatke koji se šalju ciklično ili prilikom svake promene od provajdera ka pretplatniku. Uvek se javljaju u grupama (*eventgroup*).

• Polja su kombinaciju jednog ili više od ponuđenih: *notifier, getter, setter.*

3.1.2 Servisno orjentisana komunikacija

Komunikacija zasnovana na signalu (*Signal-based communication*) već se dugo koristi u poznatijim komunikacionim protokolima. Ovakva komunikacija pretpostavlja da softver neće biti modifikovan. Takođe, podaci se šalju mrežom kad god se vrednosti podataka ažuriraju. Pošiljaoca ne brine da li čvor u mreži zahteva podatke, a ovakav pristup može opteretiti čvorove podacima koji im možda nikada neće biti potrebni [3].

U slučaju servisno orijentisane arhitekture (*Service-based*), pošiljalac šalje podatke samo ako su primaocu potrebni. Stoga, u takvom aranžmanu, server mora biti obavešten o prijemnicima koji čekaju podatke.

3.1.2.1 *Service Discovery*

U pogledu servisno orijentisane komunikacije, presudna uloga SOME/IP protokola se ogleda u *Service Discovery* (SD) modulu. Ovaj modul se koristi da eksplicitno signalizuje [4]:

• Status servisne instance (dostupna ili ne). Ukoliko je dostupna, kako doći do nje.

• Mehanizam pretplate i objave (klijent se pretplaćuje na željeni sadržaj). Nakon što se klijent pretplatio na željeni sadržaj zadatak SD-a je birati događaje/polja (*events/fields*) koji su potrebni klijentu.

3.2. SOME/IP *Binding*

ara::com predstavlja apstraktni C++ API komunikacionog posrednog softvera u okviru AUTOSAR platforme. U slučaju kada se SOME/IP koristi kao transportni sloj u arhitekturi ara::com-a uočljiva su tri dela: ara::com *frontend* koji je vidljiv korisnicima posrednog softvera uključujući *skeleton* i *proxy*. Drugi deo jeste *backend* i odnosi se na implementaciju same konekcije između modula *(binding-specific*). Treći deo jeste SOME/IP *daemon* [5].

Prva dva dela se integrišu u svaku aplikaciju posebno, dok je SOME/IP *daemon* odvojen i može biti korišćen od strane više različitih aplikacija. SOME/IP *daemon* je odlika ECU i predstavlja posrednika za svu komunikaciju.

SOME/IP *binding* ima neke od sledećih karakteristika [5]:

• Aplikacija koja koristi SOME/IP *binding* zahteva da se SOME/IP *daemon* pokrene

• SOME/IP *daemon* ne pruža podršku za monitoring mreže, pa se može desiti da *Service Discovery* uđe u inicijalnu fazu čekanja *(Initial Wait Phase*) čak i ako je veza prekinuta

• Za sve SOME/IP servise, komunikacija se obavlja preko IP soketa. Ovo znači da SOME/IP *daemon* uvek koristi kompletni mrežni stek za komunikaciju. IP poruke će biti poslate i primljene nazad od strane SOME/IP *daemon*-a. Sa konfiguracijske tačke gledišta, u slučaju servisa koji koriste SOME/IP, ne bi trebalo da postoje razlike bilo da su aplikacije u istoj ili u različitim ECU.

• SOME/IP *daemon* dodeljuje memoriju na zahtev.

4. KONCEPT REŠENJA

S obzirom na to da je cilj ovog rada simulirati SOME/IP komunikaciju na B2 ploči koja predstavlja elektronsku kontrolnu jedinicu, koncept rešenja je sadržan u sledećem:

• Izabrati dve od postojećih softverskih komponenti u datom steku i u okviru jedne napisati novu klijentsku, a u okviru druge novu serversku aplikaciju u skladu sa već postojećim servisima

• Napraviti biblioteke i integrisati ih sa datim softverskim komponentama, a zatim integrisati biblioteke u postojeći projekat

• Obezbediti podršku za komunikaciju sa pločom

• Rešenje verifikovati razmenom poruka između klijenta i servera i to ispisivanjem poruka na konzoli

5. REALIZACIJA REŠENJA

U skladu sa servisima koji već postoje u steku, napravljen je novi servis - *NewService* sa ulogom klijenta i *NewServerService* sa ulogom servera.

5.1. Konfiguracija ARXML-fajlova

S obzirom na to da se svi servisi koji se nalaze u datom steku, generišu na osnovu ARXML-fajlova koji detaljno opisuju dati servis, u svrhu dodavanja klijenta i servera, prethodno je bilo potrebno napraviti za svaki od njih novi ARXML-fajl, pri čemu su ovi fajlovi napravljeni ručno po uzoru na ARXML-fajlove drugih servisa. Podešavanje parametara u okviru ARXML-fajlova vrši se na taj način da klijent i server budu kompatibilni, odnosno da se međusobno prepoznaju i komuniciraju, u čemu se i ogleda kompleksnost zadatka. Kada se govori o prepoznavanju klijenta i servera, u slučaju SOME/IP protokola tu ulogu izvršava *Service Discovery* modul.

• Na serverskoj strani postoje tri parametra koja definišu različite faze SOME/IP *Service Discovery* protokola: *Initial Wait Phase, Repetition Wait Phase, Main Phase.*

•Ukoliko su klijent i server kompatibilni, odnosno, ukoliko klijent pronađe servera, pretplaćuje se na njegov sadržaj (*SubscribeToEvent*) i prima podatke koje je server poslao još u početnoj fazi.

Prilikom konfiguracije klijenta i servera, parametri od ključnog značaja su bili: *service id, instance id, major version* i *minor version*. Da bi servis koji nudi server odgovarao onom koji klijent traži, navedeni parametri treba da zadovoljavaju uslove *matches_service* (1) funkcije, gde su servisi čija se kompatibilnost proverava označeni sa self i other:

```
def matches_service(self, other: Service) -> bool:

    if self.service_id != other.service_id:
        return False
    if (
        self.instance_id != 0xFFFF
        and other.instance_id != 0xFFFF
        and self.instance_id != other.instance_id                    (1)
    ):
        return False
    if (
        self.major_version != 0xFF
        and other.major_version != 0xFF
        and self.major_version != other.major_version
    ):
        return False
    if (
        self.minor_version != 0xFFFFFFFF
        and other.minor_version != 0xFFFFFFFF
        and self.minor_version != other.minor_version
    ):
        return False

    return True
```

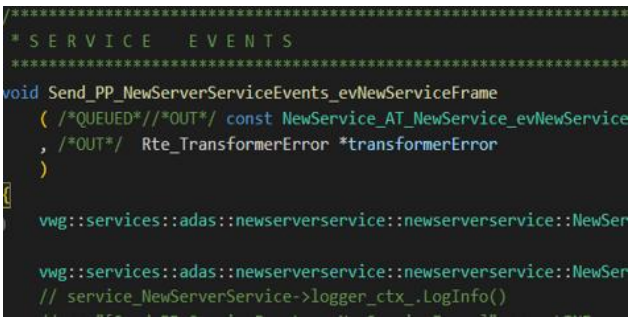
Na osnovu navedenog može se zaključiti da će klijent i server biti kompatibilni ako imaju jednake vrednosti *service id* parametra i ukoliko su vrednosti za *instance id, major version* i *minor version* ili jednake ili, bilo na klijentskoj, bilo na serverskoj strani postavljene na neku od džoker vrednosti – kada je u pitanju *instance id* to je 0xFFFF, za *major version* to je 0xFF, a za *minor version* to je 0xFFFFFFFF. U Tabeli 1. prikazane su vrednosti koje su dodeljene klijentu i serveru. Treba napomenuti da se do prethodnih zaključaka došlo eksperimentalnim putem, odnosno, primenjene su različite kombinacije vrednosti konfiguracionih parametara pre nego što je dobijena odgovarajuća kombinacija. Pri istraživanju je korišćena brojna dokumentacija iz AUTOSAR i ara::com standarda.

Tabela 1. *Konfiguracioni parametri*

POLJE	KLIJENT	SERVER
service id	24748	24748
instance id	65535	1
major version	255	1
minor version	4294967295	0
event id	32769	32769
event group id	32768	32769
udp port	42809	42810
TTL	3	3

5.2. Pravljenje klijentske i serverske aplikacije

Zadatak je osmišljen tako da se preko SOME/IP protokola šalje *event*. U tu svrhu, na serverskoj strani su napisane sve neophodne funkcije koje služe za ponudu i slanje *event*-a (Slika 1). Analogno tome, na klijentskoj strani su napisane neophodne funkcije koje služe za potražnju servisa, pretplaćivanje na servis i primanje sadržaja. U funkcije su dodati ispisi u cilju praćenja toka komunikacije kada se aplikacije spuste na ploču i mogućnosti ispravljanja eventualnih nepravilnosti.



Slika 1 *Definicija Send funkcije na serverskoj strani koja služi za slanje event-a*

5.2.1. Verifikacija koda

Nakon što su server i klijent napravljeni, neophodna je verifikacija napisanog koda. To se postiže pravljenjem biblioteka i njihovom integracijom sa softverskim komponentama.

Dve dobijene biblioteke potrebno je integrisati u postojeći projekat i obezbediti podršku za komunikaciju sa pločom, nakon čega se može pratiti tok razmene poruka.

6. VERIFIKACIJA REŠENJA

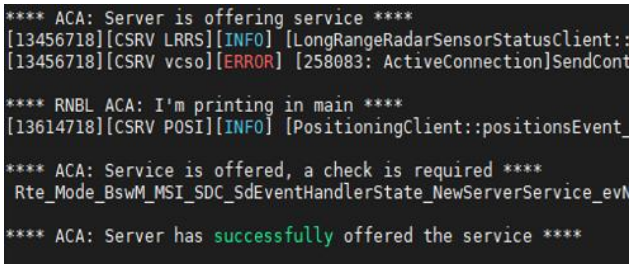
Nakon što su izvršena sva podešavanja vezana za ploču, moguće je preko *MobaXterm* konzole pratiti tok izvršavanja svih segmenata komunikacije između klijenta i servera.

Ispisi sa zvezdicama su iz testnih funkcije (jedna za klijenta, jedna za servera), dok ispisi bez zvezdica predstavljaju ispise iz samog steka.

Testne funkcije su osmišljene na taj način da se iz datog *case*-a prelazi u sledeći tek ako je ispunjen dati uslov (klijent prima servis tek ukoliko prođe proveru da se uspešno pretplatio). Na serverskoj strani testna funkcija sadrži dva *case*-a. U okviru prvog se vrši *offer*-ovanje servisa, a potom, ukoliko je ispunjen uslov da je servis uspešno *offer*-ovan, u sledećem *case*-u se vrši slanje *event*-a.

Na klijentskoj strani nulti *case* se uvek izvršava i podrazumeva traženje servera preko funkcije *StartFindService*(). U sledeći *case* se prelazi pod uslovom da je pronađen server, i u tom slučaju vrši se *subscribe*-ovanje. Ukoliko se klijent uspešno *subscribe*-ovao, prelazi se u naredni *case* gde klijent prima *event* preko funkcije *Receive*().

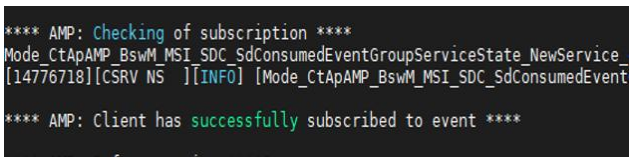
Tok dešavanja na serverskoj strani prikazan je na Slici 2. Kao što je prikazano, server je uspešno ponudio servis. Nakon toga je poslao podatke, čekajući klijenta koji će moći da ih primi. Treba napomenuti da se prethodno navedeno dešava u više ciklusa, odnosno, server konstantno nudi servis čekajući klijenta koji će da se pretplati i primi podatke.



Slika 2 *Server nudi servis preko Offer funkcije*

Sa druge strane, klijent kreće u potragu za serverom. Vrši se ciklična provera da li je klijent uspeo da pronađe server ili nije. Nakon uspešnog pronalaska servera, na konzoli se ispisuje poruka: *Server is found.*

Nakon što pronađe server, klijent pokušava da se pretplati. Poruka *Client has successfully subscribed to event* označava da se klijent uspešno pretplatio (Slika 3).



Slika 3 *Klijent se uspešno pretplatio na event koji je poslao server*

Nakon uspešnog *subscribe*-ovanja, klijent može da primi podatke od servera pozivajući funkciju *Receive*(). Da bi se ispitala funkcionalnost komunikacije potrebno je inače praznu strukturu koja predstavlja *event*, popuniti podacima čiji je tip definisan u *.typedef* fajlu.

7. ZAKLJUČAK

SOME/IP komunikacija kao relativno novo rešenje se ne može svrstati u oblasti koje počivaju na konačnim i strogim principa kao što bi to bio slučaj sa nekim starijim i duže vremena primenjivanim rešenjima iz bilo koje oblasti, jer u slučaju ovog rešenja tek mnogo toga treba da se istraži i postoji dosta prostora za otkrivanje i improvizaciju.

Sama činjenica da je u ovom radu dokazano da je moguće koristiti SOME/IP protokol za uspostavljanje komunikacije u okviru iste ECU iako se ovaj protokol uglavnom koristi za komunikaciju između dve različite ECU, daje podstrek da se ovom rešenju u budućnosti pokloni još više pažnje i da se ispitaju sve druge mogućnosti. Takođe, dobijeni rezultati pokazuju da i u okviru steka koji je korišćen postoji dosta prostora za razvoj i unapređivanje, pa s tim u vezi, može se reći da je glavni cilj ovog rada postignut, pri čemu se ostavlja prostora za dalje unapređivanje performansi, kako SOME/IP komunikacije, tako i steka.

8. LITERATURA

[1] V.I. Utkin, “Variable structure control systems with sliding modes”, *IEEE Trans. Automat. Control*, Vol. AC-22, pp. 210-222, April 1977.

[2]https://www.scribd.com/document/450637862/SOME-IP-Intro [Accessed 04 07 2021]

[3] https://www.embitel.com/blog/embedded-blog/how-some-ip-enables-service-oriented-architecture-in-ecu-network [Accessed 05 07 2021]

[4]https://www.autosar.org/fileadmin/user_upload/standards/foundation/10/AUTOSAR_RS_SOMEIPServiceDiscoveryProtocol.pdf [Accessed 29 08 2021]

[5] Technical Reference Communication Management, Vector [Accessed 10 10 2021]

Kratka biografija:

Jelena Stjepanović rođena je u Zvorniku 1997. god. *Bachelor* rad na Fakultetu tehničkih nauka iz oblasti Telekomunikacije i obrada signala odbranila je 2020.god.
kontakt: jstjepanovic1111@gmail.com