Jedno rešenje udaljenog poziva funkcija upotrebom IPC sprege

Tomislav Mitrović, Goran Miljković, Aleksandar Trkulja

***Sadržaj* — U ovom radu je definisan protokol za daljinsko pozivanje funkcija, u daljem tekstu RPC (Remote Procedure Call) između fizički ili logički odvojenih modula ili platformi. U tekstu su predstavljeni izgled RPC komandi, tok komunikacije i mehanizmi kojima je ona omogućena, specijalni slučajevi i rukovanje greškama, testiranje rada sistema, kao i korisnička sprega, u daljem tekstu API (Application Programming Interface).**

***Ključne reči* — IPC- Inter-Processor Communication Protocol, RPC- Remote Procedure Call, UART- Universal Asynchronous Receiver/Transmitter**

I. UVOD

RPC protkol je protokol za daljinsko pozivanje funkcija. Njime se postiže apstrakcija pozivanja funkcija fizički ili logički odvojenih uređaja kao da su to lokalne funkcije. Komunikacija je dvosmerna, bez postojanja vodećeg i pratećeg učesnika u njoj. Mehanizmi kojima je ona omogućena su opisani u daljim poglavljima, kao i njena dinamika.

A. Izgled RPC komande

Izgled RPC komande korištene pri projektovanju, izvedbi i testiranju protokola je prikazan u Tabeli 1.

TABELA 1: IZGLED TEST RPC KOMANDE.

Polje	Opis	Bajt
reqType	tip komande (RPC_REQ, RPC_RES)	0
funcId	identifikator funkcije koja se poziva	1
sizeOfRet	veličina povratne vrednosti funkcije	2
noOfArg	broj argumenata funkcije	3
devId	identifikator uređaja kome se šalje komanda	4
devInst	instanca uređaja kome se šalje komanda	5
payld	ret_val size_of_arg$_1$ ... size_of_arg$_n$ arg$_1$ ... arg$_n$	6 . . .

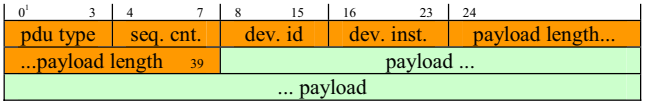
Polje *payld* je promenjive dužine i predstavlja niz bajta koji sadrže: povratnu vrednost funkcije, ukoliko ona postoji za nju je rezervisano *sizeOfRet* bajta, zatim su navedene veličine argumenata funkcije, a nakon toga i njihove vrednosti. Svi argumenti, bilo kog da su tipa i oblika (ulazni, izlazni, ulazno-izlazni), moraju biti preneseni po vrednosti, jer se radi sa odvojenim memorijskim prostorima (gde se funkcija poziva i gde se izvršava) i ne postoji mogućnost prenosa po adresi.

Uzimajući u obzir da je protokol izveden tako da bude nezavisan od oblika komande koja se šalje, uviđa se da njen predstavljeni izgled nije od suštinskog značaja za implementaciju protokola, nego je samo upotrebljen pri ovoj izvedbi, kao i za opisivanje protokola i objašnjenje procesa testiranja.

II. MEHANIZMI I DINAMIKA PROTOKOLA

A. Izgled RPC poruke i fragmentacija podataka

Za ostvarenje komunikacije između platformi je korišten IPC (Inter-Processor Communication) protokol, koji se oslanja na UART (Universal Asynchronous Receiver/Transmitter) rukovalac i razmenu podataka preko serijskog prolaza. Maksimalna količina korisnih podataka koja se može preneti jednom IPC porukom je 255 bajta. Na Sl. 1. je predstavljen izgled RPC poruke koja se prenosi jednom IPC porukom.



Sl. 1. Izgled RPC poruke

Prvih pet bajta predstavlja zaglavlje RPC poruke u kome su definisani podaci bitni za razlikovanje paketa i sledećeg su značenja:

pdu type: 4-bitna oznaka za tip poruke koja se prenosi (RPC_REQ, RPC_RES, RPC_ACK, RPC_NACK); u ovom polju se nalazi tip komande koja se šalje, dok se RPC_ACK i RPC_NACK koriste samo unutar protokola za potvrdu uspešnog, odnosno, neuspešnog prijema komande;

seq. cnt: 4-bitni brojač (modulo 16) čija vrednost predstavlja identifikator paketa (RPC poruke); obe strane poseduju svoj brojač, bez uvida u stanje brojača na drugoj strani; koristi se i za uparivanje komande/odgovora i potvrde o njenom prijemu;

Rad je delimično podržan u okviru projekta TR-6136B Ministarstva za nauku i zaštitu životne sredine Republike Srbije.

Tomislav Mitrović, Fakultet tehničkih nauka, Trg Dositeja Obradovića 6, 21000 Novi Sad, Srbija; (telefon: 381-64-9594567; e-mail: tomislav.mitrovic@micronasnit.com).

Goran Miljković, MicronasNIT, Fruškogorska 11a, 21000 Novi Sad, Srbija; (e-mail: goran.miljkovic@micronas.com).

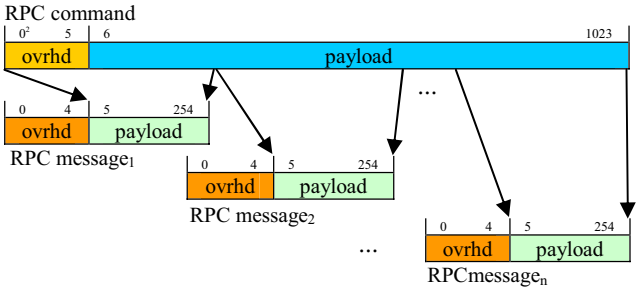
Aleksandar Trkulja, MicronasNIT, Fruškogorska 11a, 21000 Novi Sad, Srbija; (e-mail: aleksandar.trkulja@micronas.com).

[1] bit pozicije podataka unutar poruke

dev. id: 8-bitni identifikator uređaja koji šalje komandu; koristi se da bi druga strana mogla odrediti kome treba da vrati odgovor na pozvanu komandu;
dev. inst: 8-bitna oznaka instance uređaja koji šalje komandu ili odgovor; upotreba kao i kod *dev. id* polja;
payload length: 16-bitni podatak o dužini RPC komande; podatak se unosi samo u prvu RPC poruku u nizu poruka vezanih za jednu komandu, dok je u ostalim porukama ovo polje jednako nuli.

Poruka prikazana na prethodnoj slici predstavlja korisne podatke jedne IPC poruke.

Da bi se omogućilo slanje RPC komandi koje su duže od max. količine podataka koja se može preneti jednom IPC porukom, na RPC nivou je bilo potrebno uvesti podelu komande (fragmentaciju podataka) na više RPC poruka, što nije podržano IPC protokolom. Na Sl. 2. je prikazan proces fragmentacije podataka, tj. deljenja RPC komande na više RPC poruka koje se šalju jedna za drugom.



Sl. 2. Fragmentacija podataka

Za pomeranje kroz bafer komande se koristi vrednost koju vrati funkcija za slanje RPC poruke. Naime, ukoliko je poruka uspešno poslana, ta vrednost predstavlja broj poslanih bajta, a ukoliko dođe do neuspešnog slanja poruke, dolazi do prekida slanja komande uz povratnu informaciju onome ko je komandu poslao.

Na prijemnoj strani se izvršava obrnut proces, uz otkrivanje mogućih grešaka pri prenosu. Naime, paketi se prihvataju i ponovo slažu u jednu celinu, tj. RPC komanda se ponovo ujedinjuje.

Po prijemu prve u nizu poruka jedne RPC komande očitavaju se dužina komande, oznaka i instanca uređaja i pomoćni brojač paketa se postavlja na vrednost koja se nalazi u zaglavlju poruke, dok se po prijemu svih ostalih poruka proveravaju vrednosti brojača paketa i polja koje nosi podatak o dužini RPC komande u zaglavlju poruke.

Vrednosti brojača, koje stižu u zaglavljima poruka, moraju biti sekvencijalne, inače bi došlo do greške u prenosu i gubljenja paketa. Ovo svojstvo brojača paketa je upotrebljeno za otkrivanje greške.

Po prijemu svake poruke umanjuje se vrednost promenjive u kojoj se čuva dužina komande za broj primljenih bajta date poruke. Poruke se primaju sve dok je ta vrednost veća od nule ili dok ne dođe do isteka vremena predviđenog za prijem komande, koje je definisano pri pozivu funkcije za slanje.

Kako je podatak o dužini komande ubačen samo u zaglavlje prve poruke u nizu, pomoću njega je, takođe, moguće otkriti greške prenosa. Ukoliko se otkrije nepravilnost u vrednostima polja koja nose podatak o dužini komande to znači da je došlo do greške u prenosu i ona se prijavljuje drugoj strani.

B. *Retransmisija RPC poruka*

Retransmisija poruka je još jedan od mehanizama uvedenih u RPC protokolu koji nije podržan IPC-om. Naime, pokušava se izvršiti slanje RPC poruke sve dok ono ne bude uspešno izvedeno ili dok ne istekne vreme predviđeno za to. Vreme za slanje jedne RPC poruke je definisano konstantom pri implemetaciji protokola. Istom konstantom je definisano i vreme koje ograničava čekanje na njen prijem.

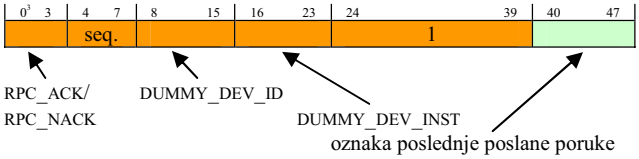
Ukoliko u predviđenom vremenskom okviru dođe do uspešnog slanja poruke, vraća se broj poslanih bajta RPC komande (ne računajući u to bajte zaglavlja RPC poruke nego samo korisne podatke), dok se u suprotnom prekida slanje komande i vraća vrednost -1, kao znak njenog neuspešnog slanja.

Na prijemnoj strani se čita sa IPC uređaja sve dok ne dođe do uspešnog prijema poruke ili dok ne istekne predviđeno vreme. Ukoliko u predviđenom vremenu ne dođe do prijema poruke dalje primanje komande se prekida i vraća se informacija o neuspešnom prijemu podataka nakon čega se ta informacija šalje i na stranu koja je komandu poslala.

C. *Mehanizam potvrde*

Ovaj mehanizam je uveden kao potvrda uspešnog, odnosno, neuspešnog prijema samo na nivou cele RPC komande/odgovora, pošto je za pojedinačne RPC poruke to već podržano IPC protokolom.

Naime, po uspešnom, odnosno, neuspešnom prijemu komande/odgovora, prijemna strana šalje drugoj strani informaciju o tome u vidu RPC poruke, koja u polju za tip nosi oznaku RPC_ACK, odnosno, RPC_NACK. Poruke ovakvog tipa kao koristan podatak nose samo brojač paketa poslednje uspešno primljene poruke, ukoliko je došlo do uspešnog prijema, ili brojač paketa poruke pri čijem je prenosu došlo do greške. Ta oznaka paketa se koristi za uparivanje poslane komande/odgovora i pozitivne ili negativne potvrde njenog prijema na drugoj strani. Izgled potvrde je predstavljen na Sl. 3.



Sl. 3. Izgled poruke koja predstavlja potvrdu prijema

U polja rezervisana za identifikator i oznaku instance uređaja koji šalje potvrdu, unutar zaglavlja poruke, postavljaju se konstante DUMMY_DEV_ID i DUMMY_DEV_INST, koje označavaju da je poruka poslana interno, ta polja su ista bez obzira koji uređaj ju je poslao.

Tek nakon prijema pozitivne ili negativne potvrde o prijemu komande na drugoj strani, strana koja je poslala komandu obaveštava korisnika o njenom uspešnom,

[2] oznake bajta unutar RPC komande i poruka

[3] bit pozicije podataka u poruci

odnosno, neuspešnom slanju. Pri tome se, po prijemu pozitivne potvrde, prelazi na čekanje odgovora na poslanu komandu. Tek nakon njegovog prijema se korisniku vraća informacija o slanju komande, ujedno i o prijemu odgovora na nju.

D. Lista komandi i parsiranje komandi/ odgovora

Unutar protokola postoji definisana lista RPC komandi u koju se smeštaju deskriptori poslanih komandi koje čekaju na prijem odgovora. Izgled deskriptora komande je predstavljen u Tabeli 2.

TABELA 2: IZGLED DESKRIPTORA KOMANDE.

Polje	Opis	Vel.
devId	identifikator uređaja koji šalje komandu	1
devInst	instanca uređaja koji šalje komandu	1
cmdId	identifikator funkcije koja se poziva	2
timestamp_ms	trenutak slanja komande	4
timeout_ms	vreme čekanja na odgovor	2

Nakon uspešnog slanja komande dolazi do upisivanja njenog deskriptora u listu komandi sa definisanim trenutkom njenog slanja i vremenom za čekanje na odgovor. Zbir vrednosti polja *timestamp_ms* i *timeout_ms* određuje vremenski trenutak do koga se čeka na odgovor. Lista komandi se periodično pregleda i proverava se validnost deskriptora u njoj. Naime, pri svakom prolazu kroz listu ispituje se da li je vremenski trenutak do kog se čeka na odgovor već prošao. Ukoliko je to slučaj, dolazi do izbacivanja deskriptora iz liste komandi.

Svrha uvođenja liste komandi jeste da se po prijemu odgovora prvo pretraži lista tj. da se ispita da li postoji komanda koja još uvek čeka na primljeni odgovor. Na osnovu rezultata provere se primljeni odgovor prosleđuje, odnosno, ignoriše.

Kao što je već pomenuto, unutar protokola postoji definisana funkcija koja ima uvid u izgled i strukturu RPC komandi koje se protokolom šalju. To je funkcija *rpcParser(...)*. Promena strukture komande za sobom povlači i moguću potrebu izmene obrade podataka unutar ove funkcije. U stvari, moguća je potreba za promenom makroa koji su definisani pri implementaciji protokola, a služe za izvlačenje oznake uređaja kome je komanda poslana i oznake pozivane funkcije.

Parser ima višestruku ulogu pri slanju i prijemu komandi i odgovora. Naime, pri slanju RPC komande on ubacuje deskriptor u listu, uz određivanje trenutka njenog slanja, a pri prijemu odgovora proverava korespondentnost oznake uređaja koji prima odgovor sa oznakom uređaja kome je odgovor poslan i proverava listu komandi, vraćajući informaciju o tome.

Parser proverava i argumente poziva funkcije za prijem komandi i odgovora.

E. Dinamika protokola

Prvi korak pri upotrebi RPC protokola jeste njegova inicijalizacija. U procesu inicijalizacije protokola dolazi do stvaranja niti, inicijalizacije semafora i vremenske kontrole, stvaranja kružnog bafera u koji se smeštaju pristigle komande i odgovori, inicijalizacije liste komandi i otvaranja IPC uređaja. Ukoliko bilo koji od koraka bude neuspešno izvršen inicijalizacija se prekida, uz odgovarajuću povratnu informaciju.

Operacije slanja i prijema komandi i odgovora su neprekidne i zaštićene semaforom, kao međusobno isključive. Naime, dok se šalje jedna RPC komanda nije moguće započeti slanje druge komande ili odgovora ili njihov prijem sa druge strane. Postoji slična zaštita i na nivou RPC poruka.

Pored toga, operacije slanja i prijema komandi, odgovora i poruka su implicitno, odnosno, eksplicitno vremenski ograničene, za šta su upotrebljene vremenske kontrole i njihove «callback» funkcije.

Operacija slanja komande je blokirajuća. Naime, nit koja je poslala komandu biće zaustavljena do prijema odgovora ili do isteka predviđenog vremena.

Po pozivu funkcije za slanje RPC komandi, dolazi do deljenja komande, ukoliko je to potrebno. Komanda je definisana kao argument poziva ove funkcije, a proces fragmentacije podataka je već opisan u poglavlju II A. Nakon slanja svih poruka vezanih za jednu RPC komandu, dolazi do čekanja na potvrdu prijema sa druge strane (dok je potvrda prijema pojedinačnih RPC poruka sadržana u povratnoj vrednosti funkcije za slanje preko IPC-a). Po prijemu potvrde, ukoliko je pozitivna, prelazi se na čekanje na odgovor na poslanu komandu.

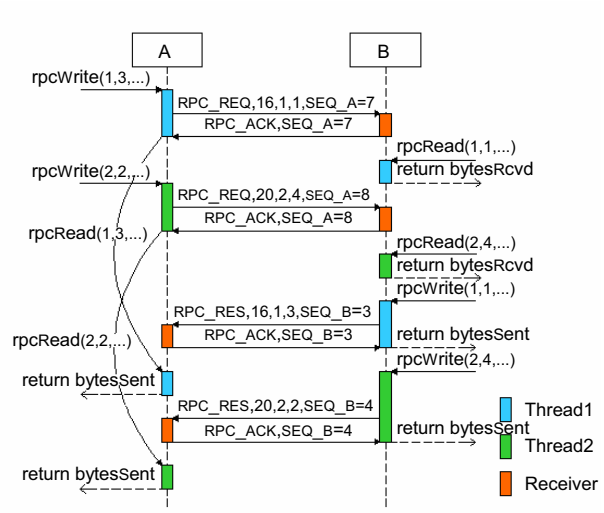
Za prijem komandi i odgovora je zadužena zasebna nit unutar protokola. Naime, ta nit predstavlja jednostavan automat sa konačnim brojem stanja (status RPC veze). Po uspešnoj inicijalizaciji RPC-a, automat prelazi u stanje u kojem periodično poziva internu funkciju za prijem komandi. Po prijemu, komanda/odgovor bivaju smešteni u kružni bafer. Pri prijemu komande, prva dva bajta u baferu u koji se ona smešta, predstavljaju oznake uređaja koji je komandu poslao, a smešteni su tu da bi aplikativni nivo znao kome treba da vrati odgovor na primljenu komandu.

Potvrda prijema se prihvata na mestima odakle se komanda šalje, dok se ona šalje iz niti za prijem komandi (sa oznakama DUMMY_DEV_ID i DUMMY_DEV_INST). Ukoliko dođe do prijema potvrde unutar niti zadužene za prijem komandi/odgovora, ona će biti ignorisana.

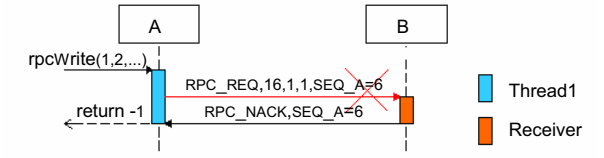
Postoji mogućnost slanja komandi iz više i ka više različitih uređaja, kao i prijem odgovora, dok se prijem komandi i slanje odgovora izvršava sa samo jednog mesta, niti unutar aplikacije koja je zadužena za prijem komande, izdvajanje potrebnih informacija, pozivanje lokalne funkcije uređaja, pripremanje i slanje odgovora. Ova funkcionalnost protokola je omogućena tako što su za svaki od uređaja definisane oznake (*device id* i *device instance*), koje se prenose unutar komande, odnosno, odgovora. Za uparivanje komandi i odgovora na njih je zadužen interni parser protokola.

Pored internog parsera, za punu funkcionalnost unutar korisničke aplikacije je potrebno napraviti eksterni parser i funkciju za pripremu komande koja će biti poslana, ali taj deo je ostavljen za aplikativni nivo. Uloga pakera komande jeste popunjavanje zaglavlja i priprema korisnih podataka RPC komande i njeno prosleđivanje funkciji za slanje komandi preko RPC-a, dok eksterni parser ima ulogu da, po prijemu komande od niti zadužene za to, izdvoji potrebne podatke, pozove odgovarajuću lokalnu funkciju, zatim pripremi i pošalje odgovor.

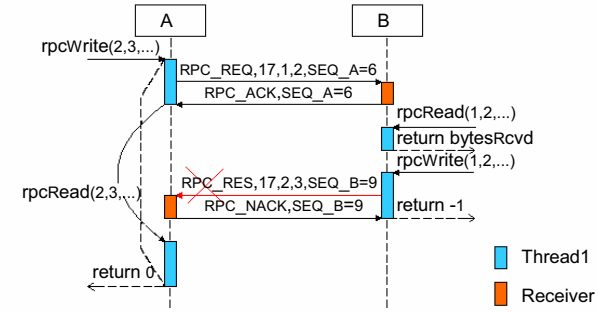
Tok komunikacije u par specijalnih slučajeva je opisan na Sl. 4. do Sl. 7.



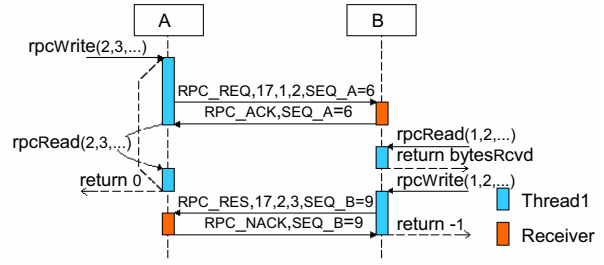
Sl. 4. Uspešno slanje i prijem dve komande i odgovora unutar zadatih vremenskih okvira; povratne vrednosti funkcija za slanje i prijem komandi predstavljaju broj poslanih, odnosno, primljenih bajta.



Sl. 5. Neuspešno slanje komande; funkcija za slanje, po prijemu negativne potvrde, vraća -1.



Sl. 6. Slučaj neuspešnog prijema odgovora na poslanu komandu; povratna vrednost funkcije za slanje je 0, jer je komanda uspešno poslana, ali nije primljen odgovor.



Sl. 7. Odgovor na poslanu komandu nije stigao u zadatom vremenskom okviru; funkcija za slanje vraća vrednost 0; naknadno primljeni odgovor biva ignorisan; uređaj koji je odgovor slao dobija negativnu potvrdu sa druge strane pa se, na osnovu te informacije, mogu poništiti efekti već pozvane lokalne funkcije.

III. PROGRAMSKA KORISNIČKA SPREGA - API

API protokola sadrži tri funkcije prikazane u Tabeli 3.

TABELA 3. OPIS PROGRAMSKE KORISNIČKE SPREGE.

rpcInit:	funkcija za inicijalizaciju IPC i RPC protokola
pov. vrednost: *int32_t*	< 0 - neuspešno >= 0 - uspešno
parametri:	nema
rpcWrite:	funkcija za slanje komandi i prijem odgovora
pov. vrednost: *int32_t*	=-1 - neuspešno slanje komande =0 - uspešno slanje, ali bez prijema odgovora >0 - uspešno slanje komande i prijem odgovora (predstavlja broj poslanih bajta)
parametri: *uint8_t myDevId* *uint8_t myDevInst* *uint8_t callType* *uint8_t* buff* *uint16_t len* *uint16_t timeout*	 - identifikator uređaja koji šalje komandu - instanca uređaja koji šalje komandu - tip komande (RPC_REQ, RPC_RES) - bafer u kome je komanda/odgovor za slanje - dužina komande/odgovora - vreme za slanje komande i prijem odgovora
rpcRead:	funkcija za prijem komandi
pov. vrednost: *int32_t*	≤0 - neuspešan prijem komande >0 - uspešan prijem (broj primljenih bajta)
parametri: *uint8_t myDevId* *uint8_t myDevInst* *uint8_t* callType* *uint8_t* buff* *uint16_t timeout*	 - identifikator uređaja koji prima komandu - instanca uređaja koji prima komandu - tip komande (RPC_REQ, RPC_RES) - bafer u koji se smešta komanda - vreme čekanja na prijem komande

IV. TESTIRANJE

Test aplikacija je razvijena za MHS_Box platformu sa MDE-B procesorom na sebi. U procesu testiranja su korištene dve pomenute platforme, koje su međusobno povezane ukrštenim serijskim kablom. Platforme su povezane i sa računarom putem serijskog kabla pa je u «starter» konzoli moguće pratiti i delimično kontrolisati izvršavanje aplikacija.

Za testiranje protokola je definisan i oblik RPC komande, kao i funkcije za pripremu, slanje i prijem komandi. Funkcija za pripremu se poziva iz uređaja koji šalje komandu i ona predstavlja paker komande. Poziv funkcije za slanje je posredan, kroz paker. Funkcija za prijem komandi se periodično poziva iz glavne programske niti. Ona je i eksterni parser RPC komandi i odgovora.

Pored ovih funkcija, definisano je i niz test funkcija koje predstavljaju lokalne funkcije uređaja. Podržane su različite vrste, one čiji je rezultat povratna vrednost (različitih tipova podataka), zatim funkcije koje rezultat vraćaju preko ulazno-izlaznih i/ili izlaznih argumenata, takođe, različitih tipova i veličina, ili na oba načina.

Za simulaciju različitih urađaja su definisane test niti, iz kojih se periodično šalju komande, pri čemu svaka od njih ima svoj jednoznačni identifikator (par *dev_id* i *dev_ inst*).

ABSTRACT

This paper defines a protocol for remote procedure calling (RPC) between physically or logically separated modules or platforms. The text contains explanation of the RPC command look, communication dynamics and mechanisms for enabling it, special cases and error handling, system testing and its programming interface.

A SOLUTION FOR RPC OVER IPC INTERFACE

Tomislav Mitrović, Goran Miljković, Aleksandar Trkulja