mr. sc. Ivan Livaja[1]
Jerko Acalin, dipl. ing.[1]

EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL (XMPP)

Stručni rad / Professional paper
UDK 004.004.1

Broj pametnih uređaja sve više raste zbog dodatnih funkcionalnosti mobilnih operacijskih sustava. Time je započeo rast u razvoju aplikacija, koje mogu pokretati takvi operacijski sustavi među kojima prednjači operacijski sustav Android. Glavna funkcionalnost mnogih od tih aplikacija je komunikacija između korisnika koristeći internet vezu. Budući da je korisniku pri komunikaciji porukama najvažnije da poruka do drugog korisnika dođe što brže, aplikacije za komunikaciju porukama obično nude uslugu trenutnog poručivanja. Trenutno poručivanje obično uključuje tri usluge: prikaz korisnikovih kontakata, prikaz prisutnosti kontakata, te razmjenu poruka između dva korisnika. Pritom se korisniku razmjena poruka i informacija o prisutnosti čini gotovo trenutnom. Navedeno postižemo koristeći XMPP - Extensible Messaging and Presence Protocol (XMPP) koji je komunikacijski protokol, te se temelji na tehnologijama otvorenog koda za brzo slanje poruka, slanja obavijesti, obavljanje video poziva i sl.. Razvijen je na Jabber tehnologiji koja omogućava otvoren način zatvorenih servisa za brzo slanje poruka. U radu su opisana osnovna svojstva protokol XMPP-a važna za razumijevanje rada protokola.

Ključne riječi: *Internet stvari, XMPP, IoT.*

1. Uvod

S brzim razvojem mobilnih mreža, ljudi mogu dobiti bogate informacije putem pametnih telefona u bilo koje vrijeme i bilo gdje. U međuvremenu, tradicionalni način dobivanja informacija postaje neučinkovit. Porast suvišnih informacija prisiljava pronalazak kvalitetnijih rješenja za pristup informacija. Napredak tehnologije vidljiv je i na svim područjima svakodnevnog života, što dovodi do brže i jednostavnije razmjene informacija. Internet je postao primarni i učinkovit medij za razmjenu informacija i resursa u cijelom svijetu. Brzo pružanje relevantnih informacija vrlo je traženo od strane korisnika kako bi im pomoglo da donesu brze i pravovremene odluke. Da bi dobili određene informacije, klijenti moraju podnijeti zahtjev tj. upit, a zatim će zaprimiti odgovor koji sadrži informacije upućene od poslužitelja. Stjecanje

[1] Veleučilište u Šibeniku

relevantnih informacija o hitnom slučaju iz njegove okolice nudi donositeljima odluka bolji uvid, učinkovito utvrđivanje odgovarajuće taktike kako bi rasporedili sigurnosne resurse i osoblje. Različite situacije u različitim okolnostima zahtijevaju prikupljanje podataka iz različitih izvora. Kako je vrijeme presudan faktor u takvim situacijama, obrada i prijenos podataka u stvarnom vremenu je neophodan faktor sustava. U današnje vrijeme komunikacija se odvija svugdje oko nas npr. od mobilnih uređaji koji mogu sadržavati više senzora, GPS sustav, kamera, mikrofon, kompas. Može se vidjeti da navedeni senzori tj. mobilni uređaj može u isto vrijeme biti i proizvođač i konzument informacija. Može se zaključiti da uz obradu podataka u realnom vremenu, jako je bitna i komponenta komunikacije.

Servis za slanje trenutnih poruka IM (*Instant messaging*) je servis koji omogućava klijentima da komuniciraju jedni sa drugima u skoro realnom vremenu. On podržava različite stilove komunikacije kao što su unicast, multi-cast i mod za grupni razgovor. Velik broj protokola i okvira (*frame-work*), koji podržavaju IM (*Instant messaging*) je napravljen do danas npr. Yahoo Messenger i ICQ. Pored toga, protokol za pokretanje sesije (*Session Instant Protocol - SIP*), protokol za pokretanje sesije za trenutno slanje poruka i prisutnost (*SIP for Instant Messaging and Presence Leveraging Ex-tensions - SIMPLE*) i IM (*Instant messaging*) koriste iste funkcionalnosti. Međutim, svaki od njih ima ozbiljne nedostatke. Komercijalni proizvodi su bazirani na vlasničkim protkolima, koji blokiraju suradnju između različitih implementacija i ograničavaju daljni razvoj. SIMPLE protokol je složen, jer je izgrađen na vrhu SIP arhitekture. Da bi se otklonili ovi problemi i olakšalo korištenje IM-a pojavilo se novo rješenje. Proširivi protokol za slanje poruka i prisutnost (*eXtensible Messaging and Presence Protocol - XMPP*) je otvoreni XML protokol za slanje poruka u gotovo realnom vremenu, davanje informacija o prisutnosti i servis za primanje zahtjeva i za njihovo odgovaranje.

2. XMPP i funkcionalnosti XMPP-a

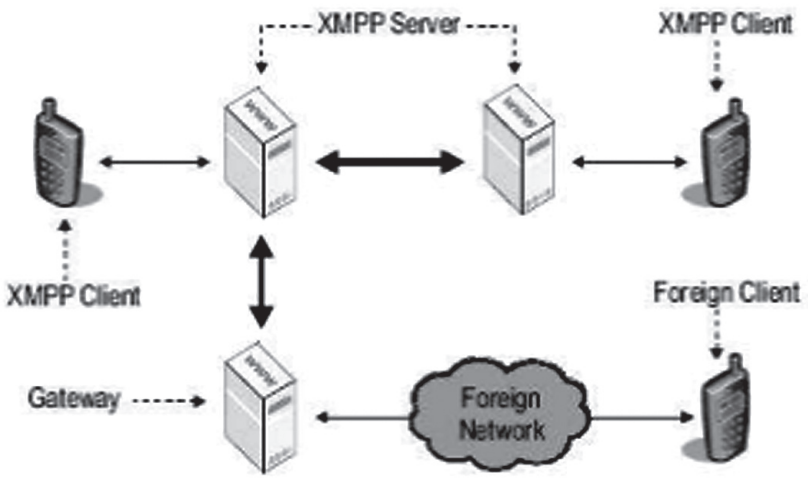
P – *Protocol*: XMPP je protokol, odnosno skup standarda koji omogućavaju međusobnu komunikaciju dvaju sustava. Široko je rasprostranjen na cijelom Internetu. P – *Presence*: indikator koji na serveru definira stanje XMPP entiteta, te spremnost entiteta na primanje poruka. M – *Messaging*: dio XMPP protokola kojeg klijent vidi: primjerice, razmijenjuje poruke između klijenata u stvarnom vremenu. X – *eXtensible*: definiran u otvorenom standardu te koristi otvoren sistem pristupa i razvoja. Drugim riječima, dizajniran je kako bi rastao i prilagođavao se protokolarnim promjenama.

Extensible Messaging and Presence Protocol (XMPP) nastao je na temelju Jabber protokola 1999. godine kojeg je kreirao g. Jeremie Miller. Budući da su poslužitelj, klijenti, programske biblioteke i dodatci za trenutno poručivanje i prisutnost bili otvorenog koda, tehnologija je brzo napredovala uz potporu zajednice programera, te je g. Miller krajem iste godine podnio zahtjev organizaciji IETF (Internet Engineering Task Force, skraćeno IETF) tražeći standardizaciju proizvoda Jabber projekta. Jabber je besplatna XML platforma otvorenog koda slična Linux-u, a služi za brzo dijeljenje poruka, videa i sličnog sadržaja u stvarnom vremenu. Nedugo nakon objavljivanja Jabber-a, g. Jeremie je dao obećavajuću potporu kako će se Jabber zajednica bazirati na IETF standardnim procesima. Za razliku od ostalih sustava brzog slanja poruka, Jabber protokol funkcionira na principu e-mail-a, odnosno koristi distribui-

ranu arhitekturu, tj. Jabber protokol dodaje sufiks za svaku adresu nakon ‹›@›› znaka, poput e-maila čime adresira servere. Takav način adresiranja omogućava Jabber serverima čitanje adresa, te se zna kako doći do drugih adresa iz različitih sustava. Jabber tehnologiju koristili su sustavi poput ICQ-a, MSN-a, AIM-a i slični sustavi za brzo slanje poruka. Njihove temeljne tehnologije formalizirane su pod imenom *Extensible Messaging and Presence Protocol* (XMPP) na IETF-u 2004. godine.

XSF (*XMPP Software Foundation),* čija je zadaća bila upravljanje projektima otvorenog koda koji su koristili tehnologije Jabber. Preko te organizacije je 2002. predan zahtjev IETF-u za standardizaciju tehnologija. Zahtjev je prihvaćen, a ubrzo je stvoren XMPP Working Group, te je pokrenuta standardizacija protokola pod imenom *Extensible Messaging and Presence Protocol*. Standardizacija je završena nakon promjena vezanih uz sigurnost poput korištenja SASL-a (*Simple Autentification and Security Layer*) za autentifikaciju i TLS-a (*Transport Layer Security*) za kodiranje kanala. Promjene su uključivale i definiranje XMPP adresa poznatih kao JID 3 (*Jabber ID*), dodatke za blokiranje i biranje kontakata. Do kraja 2004. godine IETF prihvaća XMPP i njegove standarde. Nakon standardizacije, započinje razdoblje prihvaćanja protokola XMPP i razvoja tehnologija i usluga koje se temelje na njemu. U 2005. pojavljuju se XMPP usluge Google, Talk IM i VoIP (*Voice over Internet Protocol*), te utjecajne tvrtke poput IBM, Apple, Nokia-e i sličnih počinju koristiti XMPP tehnologiju. Tijekom 2010. godine društvena mreža Facebook predstavlja XMPP podršku za svoju uslugu razmjene kratkih poruka (Chat). Facebook ne sadrži interni XMPP poslužitelj, već pruža XMPP sučelje klijentskim aplikacijama. Posljedica toga jest da je XMPP podrška ograničena, te pojedine značajke nisu dostupne. Otvoreni protokol XMPP se nadograđuje i proširuje dodatcima, a za standardizaciju i definiciju brine se XSF kao samostalna neprofitna organizacija. Tokom godina razvoja formalno je odobreno od strane IETF nekoliko internet specifikacija. Osim toga XSF radi na stalnom unapređenju ovog okvira. Izvorno dizajniran kao Instant Messaging protokol XMPP nasljeđuje proširiva svojstva XML-a. Format poruke koristi se za prijenos različitih vrsta malih poruka između proizvoljnih entiteta u gotovo realnom vremenu. Poruke se šalju putem XMPP poslužitelja tzv. JID-ovima. Bare JID obično predstavlja korisnika (npr. user@mydomain.org), dok Full JID predstavlja resurs u vlasništvu korisnika (npr. user@mydomain.org/phone). Poruke se mogu slati ili na konkretan resurs (koristeći Full JID) ili korisniku putem Bare JID-a. Kao što je iz oblika adrese vidljivo, svaki JabberID sadrži domenski dio. Domenski dio služi za određivanje poslužitelja na kojeg se klijent spaja. Pri tome se koristi sustav DNS (Domain Name System) koji služi za dobivanje IP adrese poslužitelja iz simboličkog imena. Kada se klijent povezuje na XMPP poslužitelj, na adresu se dodaje identifikator resursa za tu jedinstvenu konekciju. Taj resurs se tada koristi da bi se poruke usmjerile na točno tu konekciju umjesto na neku drugu koju je korisnik mogao otvoriti istovremeno koristeći više uređaja. Resurs se dodaje na kraj adrese, te je adresa tada oblika user@mydomain.org/phone. Resurs se obično sastoji od imena računala, lokacije ili klijentskog softvera. Stvarni izvor primanja određuje se prioritetima, koji se mogu postaviti slanjem obavijesti o prisutnosti poput e-pošte. Obično, postoji i jedan XMPP poslužitelj po organizaciji kao i neki neovisni poslužitelji koji nude besplatno ili plaćeno korištenje njihove usluge za privatne osobe. Kao što je prethodno spomenuto, ovi poslužitelji se međusobno povezuju preko DNS-a za prijenos poruka na ciljanu domenu. Njime XMPP Standards Foundation (XSF) omogućuju dodatne funkcionalnosti kao što su multi-user chat, dijeljenje datoteka i poruka bez poslužitelja.

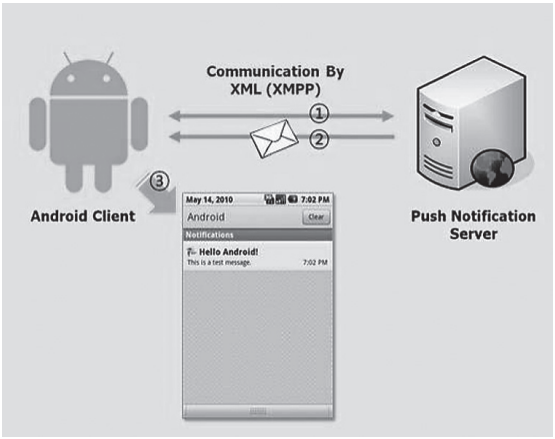
Slika 1. XMPP arhitektura



Decentralizirana klijent-poslužitelj arhitektura omogućava učinkovito razdvajanje obaveza. Poslužiteljska strana obično je zadužena za sigurnost, autentifikaciju korisnika, enkripciju podataka i slično. S druge pak strane, to omogućava programerima koji razvijaju klijentske aplikacije da se mogu više usredotočiti na korisničko iskustvo. Za razliku od Web arhitekture, u XMPP arhitekturi poslužitelji su međusobno povezani. Kada klijent šalje poruku drugome klijentu koji se nalazi na drugoj domeni, on se povezuje na svoj poslužitelj, koji se tada direktno povezuje na poslužitelj na kojem se nalazi primatelj. Poruka na taj način ne prolazi kroz posredničke poslužitelje kao što je to slučaj kod email arhitekture.

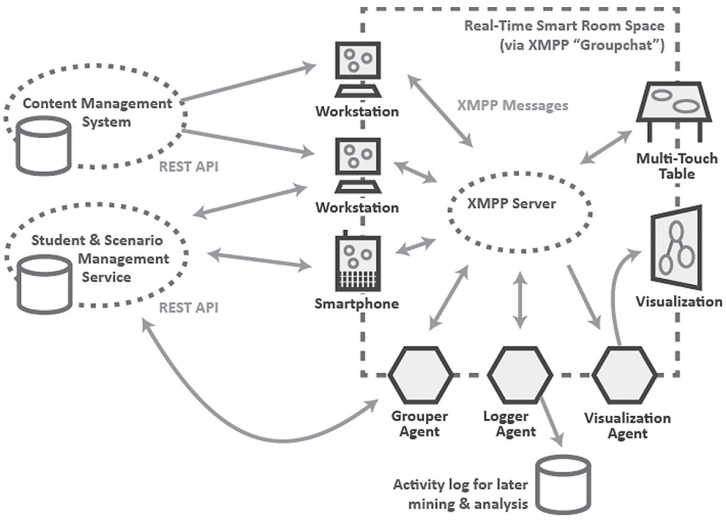
XMPP serveri pružaju mogućnosti slanja i primanja poruka i informacija o prisutnosti, te XML rutiranje. Primjeri XMPP servera su CommuniGate Pro, Apache Vysper, iChat server, Openfire, Jerry Messanger, Siemens OpenScape, Tigase, Wokkel.

Slika 2. Sustav za razmjenu poruka temeljen na XMPP



Protokol, koji se razvio iz Jabber zajednice, originalno je dizajniran kako bi omogućio otvorenu, sigurnu i decentraliziranu alternativu klijentskim servisima brzih slanja poruka. Tehnologije na kojima se temelji XMPP uključuju: osnovni XML streaming sloj, enkripciju kanala koristeći TSL protokol, strogu autentikaciju koristeći SASL protokol, UTF-8 zapis kodnih točaka uključujući internacionalizaciju adresa, ugrađene informacije o dostupnosti mreža, prisutnost upisa dvosmjerne autorizacije te prisutnost kontakt lista. Postoje mnoge implementacije jezgre XMPP-a, ali u suštini XMPP se implementira kao klijent, server ili knjižnice programskog koda.

Slika 3. Princip rada XMPP-a



Prijenos podataka temelji se na XML-u. Prilikom stvaranja sjednice sa poslužiteljem otvara se TCP konekcija te uspostavlja dvosmjerni tok podataka sa poslužiteljem. Nakon uspostave veze, klijent i poslužitelj mogu asinkrono izmjenjivati neograničen broj XML poruka, takozvanih XMPP strofa. Postoje 3 vrste strofa: <*message/*>, <*presence/*> i <*iq/*>. Strofa <message/> koristi se za slanje poruka, upozorenja, obavijesti te za grupni razgovor. Postoji pet vrsta strofe <*message/*> ovisno o atributu type: normal, chat, groupchat, headline i error. Atributi *from* i *to* sadrže JabberID pošiljatelja i primatelja poruke. Primjer strofe <*message/*> može se vidjeti na navedenom primjeru:

```
<message type="chat"
from="ivan@primjer.com"
to="marko@primjer.com">
<body>Pozdrav!</body>
</message>
```

Strofa <*presence/*> koristi se za razmjenu informacije o prisutnosti korisnika na mreži. Svaki korisnik ima mogućnost postaviti svoje stanje prisutnosti, koje drugi korisnici mogu vidjeti, ako su za to autorizirani. Autorizacija se odvija kroz jednostavan zahtjev za pretpla-

tom kojeg korisnik treba odobriti ukoliko želi da drugi korisnik vidi njegovo stanje prisutnosti. Ako je autorizacija odobrena, korisnici će redovito biti obaviješteni o promjenama stanja prisutnosti. Strofa <*iq/*> koristi se za slanje zahtjeva i odgovora kao što su zahtjev za registracijom, zahtjev za dohvatom liste kontakata i slično. Sadrži atribut type koji može poprimiti jednu od četiri vrijednosti: *get, set, result* i *error*. Vrijednost *get* koristi se za dohvat informacije, dok se *set* koristi za pružanje informacije. Vrijednost *result* koristi se kao odgovor na operaciju *get* ili potvrda operacije *set*. Vrijednost *error* služi za dojavu greške prilikom procesiranja zahtjeva. XMPP ne zahtijeva potvrdu o isporuci za sve vrste strofa. Pretpostavlja se da su strofe <*message/*> i <*presence/*> uspješno dostavljene ako nije dojavljena greška, te se ne zahtijeva potvrda primatelja. Strofa <*iq/*> s druge strane zahtijeva odgovor o uspješnom primitku poruke ili dojavu greške.

Primjer slanja poruke može se vidjeti niže:

```
ChatManager chatManager = connection.getChatManager();
Chat chat = chatManager.createChat("ja@primjer.com", new
MessageListener() {
public void processMessage(Chat chat, Message message)
{
System.out.println("Primio poruku: " + message);
}
});
Message message = new Message();
message.setBody("Pozdrav!");
message.setProperty("vrijeme","sunce");
chat.sendMessage(message);
```

XMPP se sastoji od 5 ključnih tehnologija: Core, Jingle, Multi-User Chat, PubSub i BOSH. Svaka ovih tehnologija ima svoju funkciju i svoj zadatak.

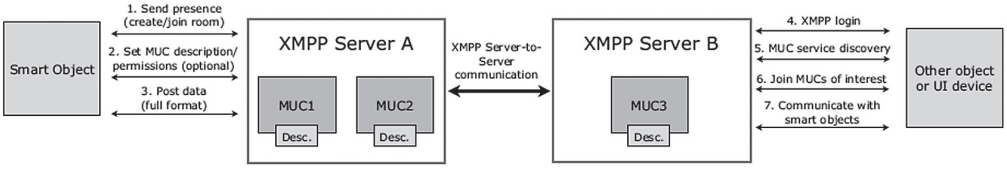
2.1. Jingle

U suštini, Jingle omogućava Jabber klijentima postavljanje, upravljanje i skidanje multimedijskih sesija. Multimedijske sesije podržavaju širok raspon primjene poput glasovnih poruka, video razgovora ili prijenosa podataka, te širok raspon metoda za prenošenje medija, kao što su TCP-a, UDP-a, RTP-a. Signal za uspostavljanje Jingle sesije šalje se putem XMPP-a, dok su mediji poslani direktno peer-to-peer metodom ili putem medijskog releja. Jingle pruža prilagodljivi okvir za sve vrste primjene i za metode prenošenja medija. Prijenos glasa i video razgovora Jingle koristi RTP za prijenos medija, čime je kompatibilan sa postojećim multimedijskim tehnologijama, poput *Session Initation Protocol* (SIP). Nadalje, značajnost Jingle signaliziranja je osmišljena kako bi Jingle uz SIP bio dosljedan i *Session Description Protocol*-u, čineći ga neposrednim za pružanje signalnih pristupnika između XMPP mreža i SIP mreža.

2.2. Multi-User Chat

Multi-User Chat (MUC) je XMPP ekstenzija za razmjenu grupnih informacija, kao kod *Internet Relay Chat-a* (IRC), gdje više korisnika može međusobno razmjenjivati poruke u privatnoj sobi ili u nekom određenom kanalu. Za razliku od standardne sobe za razgovor, ovaj protokol omogućuje kreiranje specifične teme, pozivanje drugih korisnika, te moderatorima grupe omogućuje izbacivanje ili kažnjavanje korisnika, postavljanje ulazne šifre za razgovor i drugo. Kako su MUC sobe za razgovor bazirane na XMPP-u, osim za razmjenu običnog teksta, mogu se koristiti i za širok izbor XML sadržaja.

Slika 4. XMPP Multi-User Chat



Pametni objekt se najavljuje slanjem prisutnosti u MUC sobu. Ako soba još ne postoji, stvara ga MUC produžetak automatski. Višestruki pametni objekti mogu se pridružiti istoj sobi. Stoga je moguće stvoriti prostoriju, koja predstavlja mjesto ili drugi objekt u stvarnom svijetu s višestrukim senzorima ili pametnim objektima. Vlasnik tj. pametni objekt koji ju je stvorio može promijeniti opis i dopuštenja MUC sobe. Dopuštenja se koriste za ograničavanje pristupa usluzi MUC soba zaštitom lozinkom ili pomoću crno / bijelih lista.

2.3. PubSub

PubSub je protokol koji omogućava publish-subscribe funkcionalnost. Protokol dopušta XMPP-ovim subjektima izradu čvorova, odnosno tema, na PubSub uslugama, te objavljivanje informacija na tim istim čvorovima. Objava informacija dolazi kao obavijest svim onim subjektima, koji su prijavljeni na čvorove. Stoga PubSub može poslužiti kao temelj za široku paletu aplikacija koja uključuje: primanje novosti, prijenos sadržaja, geolokaciju, sustave za upravljanje mrežom te bilo koju drugu aplikaciju koja zahtijeva primanje obavijesti. PubSub sadrži i protokol *Personal eventing protocol* (PEP) koji pruža svjesnost prisustva PubSub profila, te koji omogućava svakom korisniku funkcioniranje kao virtualna PubSub usluga za prisutnost, za društvene mreže, te za interakcije u stvarnom vremenu.

2.4. BOSH

Bidirectional-streams Over Synchronous HTTP (BOSH) je tehnologija koja omogućava dvosmjernu komunikaciju putem HTTP-a. BOSH imitira primitivne transportne protokole, poput TCP-a. Za aplikacije koje zahtijevaju ''push'' i ''pull'' komunikacije BOSH je znatno više propusniji, učinkovitiji i prilagodljiviji u odnosu na klasične dvosmjerne HTTP transportne protokole i tehnike poznate kao AJAX. Do danas, BOSH je korišten uglavnom za transport prometa

između Jabber i XMPP klijenata ili servera. Kako bilo, BOSH nije vezan isključivo za XMPP, već može biti korišten i za druge vrste prometa.

3. Cloud messaging

3.1. Google Cloud Messaging

Google Cloud Messaging (GCM) je besplatni servis koji omogućava programerima slanje poruka između servera i klijentskih aplikacija. Slanje poruka uključuje slanje sa servera na klijentske aplikacije, ali i slanje poruka od aplikacija prema serverima. Poslane poruke dolaze u obliku obavijesti.

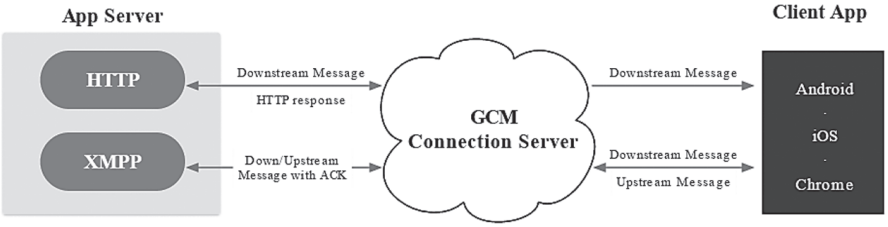
Slika 5. GCM arhitektura

GCM arhitektura se sastoji od sljedeće tri komponente:

– GCM *Connection Server* - server koji preuzima poruke sa aplikacijskog servera i šalje ih na klijentsku aplikaciju, koja se pokreće na klijentskom uređaju. *Connection Server* pruža Google i to servere za HTTP i XMPP protokole. Na aplikacijski server, može se koristiti zasebno HTTP ili XMPP, ili se mogu koristiti skupa.

– *Application Server* - server kojeg je nužno implementirati u programsko okruženje. *Application Server* šalje podatke prema klijentskoj aplikaciji putem odabranog GCM poslužiteljskog servera koristeći odgovarajući XMPP ili HTTP protokol. Prije nego što programiramo klijentsku aplikaciju koja koristi GCM, aplikacijski server bi trebao izvršavati sljedeće naredbe: komunicirati sa klijentom, slati pravilno formatirane zahtjeve prema GCM poslužitelju, upravljanje poslanim zahtjevima te vraćati iste zahtjeve, sigurno pohraniti ključ servera i klijentski registracijski token, te za XMPP protokole server treba generirati jedinstveni ID poruke za jednostvniju identifikaciju poruka.

Slika 6. GCM arhitektura

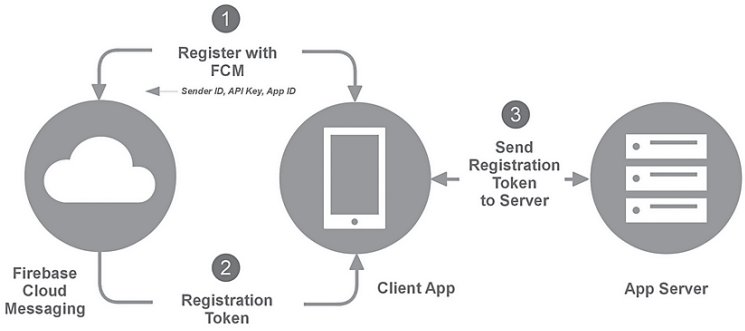


– Klijentska aplikacija – aplikacija koja se nalazi na uređaju klijenta. Kako bi mogla slati i primati poruke prema i od GCM-u, treba biti registrirana na GCM čime joj se pridružuje jedinstveni identifikator, poznat pod nazivom registracijski token.

3.2. Firebase Cloud Messaging

Ukoliko se pretpostavi, da je aplikacija klijenta skupni članak s više suradnika, svatko bi trebao biti u mogućnosti poslati poruku kada objavljuju novi članak. Ova poruka može sadržavati URL, kako bi aplikacija klijenta mogla preuzeti članak. Umjesto da centraliziramo svu aktivnost slanja na jednom mjestu, FCM (*Firebase Cloud Messaging*) nam daje mogućnost da svaki od tih suradnika pošalje vlastite poruke.

Slika 7. FCM arhitektura



FCM (*Firebase Cloud Messaging*) omogućuje širok raspon slanja poruka. Moguće je slati dvije vrste poruka klijentima: Poruke obavijesti ili poruke podataka koje obrađuje aplikacija klijenta. Poruke obavijesti sadrže unaprijed definiran skup korisnički vidljivih ključeva. Poruke podataka, za razliku od njih, sadrže samo prilagođene parove ključ / vrijednost. Poruke obavijesti mogu sadržavati opcionalno dodatne podatke, koje se isporučuju kada korisnici konzumiraju obavijest. Da bi poslali poruke obavijesti pomoću administratorskog SDK-a ili FCM protokola, potrebno je postaviti obavijest s potrebnim unaprijed definiranim skupom opcija ključ / vrijednost za korisnički vidljivi dio poruke obavijesti. Na primjeru dolje može se vidjeti poruka obavijesti u obliku JSON u aplikaciji za IM. Korisnik može očekivati da će vidjeti poruku s naslovom «Sport» i tekstom «Hrvatska-Turska»:

```
    {
  «to» : «RNwTe322:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...»,
  «notification» : {
   «body» : «Sport»,
   «title» : «Hrvatska-Turska»,
   «icon» : «Cro»
  }
 }
```

Za slanje vrijednosti korisničkog sadržaja u aplikaciju klijenta. Poruke mogu imati maksimalno veličinu od 4KB podataka. Npr. poruka JSON-oblikovana je u istoj aplikaciji za IM kao i prethodno, gdje su informacije obuhvaćene podatkovnim ključem.

```
    {
  "to" : "RNwTe322:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1...",
  "data" : {
   "Nick" : "Ivan",
   "body" : "Sport",
   "Room" : "Hrvatska-Turska"
  },
}
```

Također, primjer poruke obavijesti s dodatnim opterećenjem podataka može se vidjeti na navedenom primjeru niže:

```
     {
  "to" : "APA91bHun4MxP5egoKMwt2KZFBaFUH-1RYqx...",
  "notification" : {
   "body" : "Sport",
   "title" : "Hrvatska-Turska",
   "icon" : "Cro"
  },
  «data» : {
   «Nick» : «Ivan»,
   «Room» : «Hrvatska-Turska"
  }
 }
```

Postoje dvije mogućnosti za dodjeljivanje prioriteta isporuke poruka: normalni i visoki prioritet. Poruke normalnog prioriteta neće otvoriti mrežne veze na uređaju za npr. vrijeme spavanje i njihova isporuka može biti odgođena da bi se baterija očuvala. Za manje vremenski osjetljive poruke, kao što su obavijesti o novoj e-pošti ili drugim podacima za sinkronizaciju, mogu se koristiti uobičajeni prioritet prikazivanja. Visoki priorite je zadani prioritet za poruke obavijesti. FCM tada pokušava odmah isporučiti poruke s visokim prioritetom, omogućujući FCM uslugu da probudi uređaj za vrijeme spavanja kada je to moguće i otvori mrežnu vezu s poslužiteljem aplikacije. Na primjer, aplikacije s upozorenjima za razmjenu trenutačnih poruka, chata ili glasovnih poziv obično trebaju otvoriti mrežnu vezu i pobrinuti se da FCM bez odgađanja isporučuje poruku uređaju. FCM obično dostavlja poruke odmah nakon slanja. Međutim, to

možda nije uvijek moguće. Npr. ukoliko je uređaj isključen, izvan mreže ili na drugi način nedostupan. FCM može namjerno odgoditi isporuku poruke, kako bi spriječio aplikaciju da konzumira prekomjerne resurse i negativno utječe na trajanje baterije. Kada se to dogodi, FCM pohranjuje poruku i isporučuje je čim to bude moguće. Iako je to u redu u većini slučajeva, postoje neke aplikacije za koje se kasna poruka nikad ne bi mogla isporučiti. Npr. ako je poruka poziv na događaj, beskorisno je ako se poruka primi nakon završetka događaja. Može se koristiti parametar *time_to_live*, podržan u oba HTTP i XMPP zahtjeva, da bismo odredili maksimalni vijek trajanja poruke. Vrijednost ovog parametra može biti od 0 do 2,419,200 sekundi (28 dana), a odgovara maksimalnom vremenskom razdoblju za koji FCM pohranjuje i pokušava poslati poruku. Prednost određivanja vijeka trajanja poruke je da FCM nikada ne priguši poruke s *time_to_live* (TTL) vrijednosti od 0 sekundi. Drugim riječima, FCM jamči najveći napor za poruke koje moraju biti isporučene "trenutno". Vrijednost *time_to_live* od 0 znači da se poruke koje se ne mogu odmah isporučiti odbacuju. Ako uređaj nije povezan s FCM-om, poruka se pohranjuje sve dok se ne uspostavi veza. Kad se uspostavi veza, FCM dostavlja sve poruke koje sun a čekanju uređaju. Ako se uređaj nikad više ne poveže (na primjer, ako je tvornički resetiran), poruka koja je istekla izbacuje se iz FCM pohrane.

4. Zaključak

Protokol Jabber koji je služio za razmjenu brzih poruka, video sadržaja u stvarnom vremenu, razvio se u standard XMPP *(Extensible Messaging and Presence Protocol)* pomoću kojeg je osim brzog slanja poruka omogućena i kompletna komunikacija između razvojnih programera, servera i krajnjih korisnika aplikacija i uređaja. Takva komunikacija podrazumijeva slanje obavijesti, prijenos sa servera i prema serveru grupe podataka, stvaranje servisa kao što je *Firebase Cloud Messaging* za obavještavanjem krajnjih korisnika ili podizanje komunikacije na višu razinu razvojem Internet of Things-a. XMPP/Jabber tehnologija je postigla veliki uspjeh, pogotovo od kada je IETF odobrio osnovne protokole. To je dovelo do značajnih implementacija i velikog napretka u razvoju. Nove mogućnosti su ugrađene u okvir, što je i dovelo do velike primjene protokola. Osim komunikacija i razmjena podataka XMPP putem IoT-a mijenja sve aspekte života. XMPP cloud messaging postupak razmjene poruka se razlikuje od tradicionalnih metoda objavljivanja informacija: klijent se mora povezati i prijaviti na oblak poslužitelj, te poslati zahtjev poslužitelju. Kada je oblak poslužitelj prihvatio zahtjev, on će odgovoriti sa rezultatima na upit klijenta. Predloženi mehanizam poboljšava učinkovitost objavljivanja informacija. Nadalje, tehnologija u oblaku aktivno i selektivno grupira korisnike na temelju različitih zahtjeva, što čini mobilne mreže temeljene na računarstvu u oblaku inteligentnijim mrežama. U usporedbi s tradicionalnom mrežom, sve su prednosti neusporedive. XMPP je otvoreni protokol s velikom mogućnošću razvoja, koji je kompatibilan s različitim hardverskim i softverskim platformama. U budućnosti će se cloud server messaging široko primjenjivati u inteligentnom domu, internetu stvari, inteligentnih robota i mnogih drugih područja. Svrha IoT-a je povezati uređaje, poput senzorskih uređaja na druge uređaje, primjerice na pametne telefone, kako bi mogli upravljati istima. Ustvari, Internet of Things (IoT) je ono što se dobije kada se povežu stvari, koje nisu upravljane ljudima na Internet. Povezivanje uređaja na IoT podrazumijeva kompletno sve uređaje koji se svakodnevno koriste. Predviđa se da će do 2020. godine biti povezano oko 50 milijardi uređaja na internet.

LITERATURA

1. Gero Mühl, Ludger Fiege, and Alejandro P. Buchmann. 2002. Filter Similarities in Content-Based Publish/Subscribe Systems. In Proceedings of the International Conference on Architecture of Computing Systems: Trends in Network and Pervasive Computing (ARCS '02), Hartmut Schmeck, Theo Ungerer, and Lars C. Wolf (Eds.). Springer-Verlag, London, UK, 224-240.
2. https://xmpp.org/
3. http://searchnetworking.techtarget.com/definition/Jabber
4. http://www.forbes.com/sites/jacobmorgan/2014/05/13/simple-explanation-internet-things-that-anyone-can-understand/#25529cce6828
5. https://geek.hr/clanak/internet-of-things-iot/
6. https://developers.google.com/cloud-messaging/gmc
7. Y. Huang and H. Garcia-Molina, "Publish/subscribe in a mobile environment," Wirel. Netw., vol. 10, no. 6, pp. 643-652, 2004.
8. Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.: The Many Faces of Publish/Subscribe. ACM Computing Surveys 35 (2003) 114–131
9. Ivana Podnar Žarko, Krešimir Pripužić, Ignac Lovrek, Mario Kušek. Raspodijeljeni sustavi, Radna inačica udžbenika v.1.0 2013./2014.
10. Ala¨Kasab, Geospatial-based Publish/Subscribe: Improving Real-time Notification and Situational Awareness in Fire Emergency, University of Calgary ,2009.
11. http://www.igniterealtime.org/
12. Aris M. Ouksel, Oana Jurca, Ivana Podnar and Karl Aberer, Efficient Probabilistic Subsumption Cheking for Content-Based Publish/Subscribe Systems
13. Chin-Yuan Huang: GeoPubSubHub: A Geospatial Publish/Subscribe Architecture for the World-Wide Sensor Web, Department of Geomatics Engineering, University fo Calgary 2014.
14. Sasu Tarkoma, Publish / Subscribe Systems: Design and Principles 1st Edition
15. https://firebase.google.com/docs/cloud-messaging/concept-options
16. Peter Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Address Format. IETF RFC 6122, mart 2011. http://tools.ietf.org/html/rfc6122
17. Peter Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence. IETF RFC 6121, mart 2011. http://tools.ietf.org/html/rfc6121
18. Peter Saint-Andre. Extensible Messaging and Presence Protocol (XMPP): Core. IETF RFC 6120, mart 2011. http://tools.
19. ietf.org/html/rfc6120

Summary

EXTENSIBLE MESSAGING AND PRESENCE PROTOCOL (XMPP)

The number of smart devices is growing ever more due to additional functionalities of mobile operating systems. This has begun a growth in the development of applications that can trigger such operating systems with the predominance of the Android operating system. The main functionality of many of these applications is communication between users by using the internet connection. Since users consider the speed of message exchange as the most important aspect when communicating with messages to another user, messaging applications usually offer instant messaging service. Currently, messaging usually includes three services: displaying user contacts, displaying presence of contacts, and messaging between two users. By doing so, the user considers messages exchange and presence information almost instantaneous. This is achieved by using the XMPP - Extensible Messaging and Presence Protocol (XMPP), which is a communication protocol based on open code technologies for swift sending messages, sending notifications, performing video calls, etc. It has been developed on Jabber technology that enables open mode of closed services for swift messaging. The paper describes the basic properties of the XMPP protocol important for understanding the protocol work.

Key words: *Internet, XMPP, IoT.*