

# Uvod u IoT sustave

Darko Andročec

- Termin Internet of Things – Internet stvari – IoT smislio je Kevin Ashton s MIT-a 1999. u jednoj prezentaciji za Proctor & Gamble (P&G)
- Obećavajući koncept -> održivo tehnološko rješenje
- Tehnologija budućnosti -> svakodnevica
- Neograničena komunikacija stvari preko interneta
- Širok raspon domena primjena (pametno zdravstvo, pametni grad...)

- Things (stvari)
- Senzorske, aktuarske, pohraniteljske i procesorske mogućnosti
- Malene dimenzije, opremljeni jednim ili više senzorom, malom količinom memorije i mikroprocesorom
- Osnovni cilj IoT-a je omogućiti stvarima oko nas da osjete i povremeno prenose neke parametre okoliša
- Pametni frižider

## Pametna stvar

Fizički objekt digitalno nadopunjen jednom ili više sljedećih elemenata:

- Senzori (temperature, svjetla, pokreta itd.)
- Aktuatori (ekrani, zvuk, motori i sl.)
- Obrada (mogućnost pokretanja programa i logike)
- Sučelja za komunikaciju (žične ili bežične)

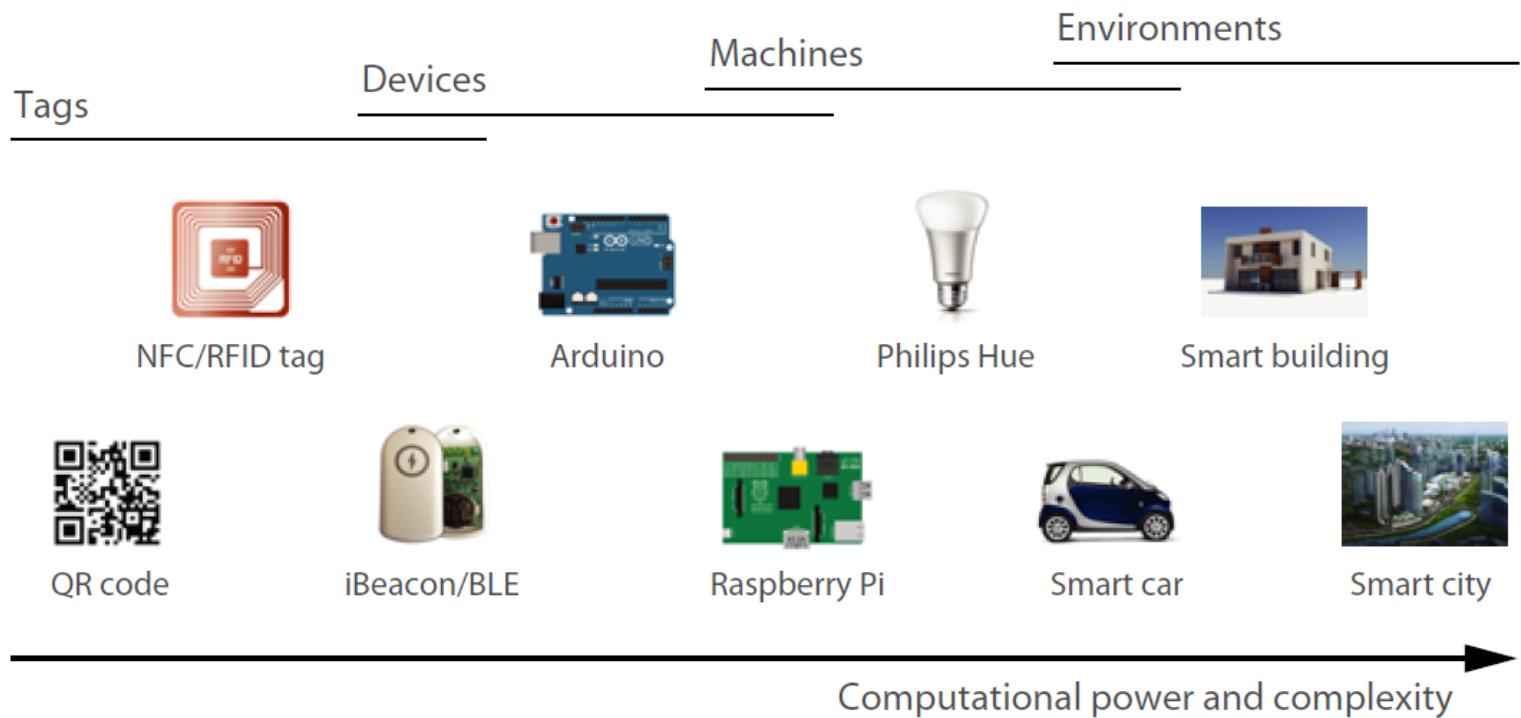
## Definicija IoT-a (1)

- „Internet stvari (IoT) je mreža namjenskih fizičkih objekata (stvari) koji sadrže ugrađenu tehnologiju za komunikaciju i osjećaju ili komuniciraju sa svojim unutarnjim stanjima ili vanjskim okolišem. Povezivanje sredstava, procesa i osoblja omogućuje dohvaćanje podataka i događaja iz kojih poduzeće može naučiti ponašanje i korištenje, reagirati preventivnim djelovanjem, povećati ili transformirati poslovne procese. IoT je temeljna sposobnost za stvaranje digitalnog poslovanja.” (Gartner)

## Definicija IoT-a (2)

- 2015 IEEE – dokument *Towards a Definition of the Internet of Things*
- „IoT se može definirati kao međusobno povezivanje nekoliko na jedinstven način prepoznatljivih, sveprisutnih i internetom povezanih stvari koje su opremljene senzorskim/aktuatorskim i komunikacijskim jedinicama te su sposobne za inteligentno donošenje odluka. Uređaji su također interoperabilni i mogu se samostalno konfigurirati i podešavati (programirati).” (IEEE)

# Krajolik interneta stvari



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Internet of Everything (IoE)
- „Internet svega (IoE) okuplja ljudе, procese, podatke i stvari kako bi mrežne veze bile relevantnije i vrijednije nego ikad prije, pretvarajući informacije u radnje koje stvaraju nove sposobnosti, bogatija iskustva i neviđeno ekonomiske prilike za tvrtke, pojedince i zemlje.” (Cisco)

- Industrial Internet of Things (IIoT)
- „IIoT je sustav koji se sastoji od umreženih pametnih objekata, cyber-fizičke imovine, povezane generičke informacijske tehnologije i optionalno oblaka ili ruba (cloud or edge), koje omogućuju inteligentno i autonomno u stvarnom vremenu pristup, prikupljanje, analizu, komunikaciju i razmjenu procesa i informacija o proizvodima i/ili uslugama, unutar industrijskog okruženja kako bi se optimizirala ukupna vrijednost proizvodnje. Ova vrijednost može uključivati: poboljšanje isporuka proizvoda ili usluge, povećanje produktivnosti, smanjenje troškova rada, smanjenje potrošnje energije i smanjenje ciklusa izrade po narudžbi.“ (Boyes et al.)

- Međusobna povezanost stvari
- Povezanost stvari na Internet
- Jedinstvena identifikacija pojedine stvari
- Sveprisutnost (*ubiquity*)
- Senzorske i aktuarske mogućnosti
- Ugrađena inteligencija
- Interoperabilna mogućnost komunikacije
- Samokonfiguriranje
- Programibilnost

## Heterogenost IoT-a

- IoT uređaji su heterogeni – funkcionalnost, veličina, trošenje energije, trošak
- Različiti senzorski parametri koji se prate – temperatura, vlažnost, količina svjetlosti, otkucaji srca, ECG i EEG signali...
- Minijaturni uređaji – veličine mobitela – pa sve do dronova
- Trošenje energije ovisi o veličini, funkcionalnosti i korištenim komponentama

- Svaki IoT uređaj ima jedinstveni identifikator (UID) za identifikaciju uređaja na internetu
- Koriste se RFID oznake, MAC i IP adrese, te brojevi mobitela i SIM-ovi
- Globalno povezivanje stvari preko interneta – geoprostorna distribuiranost IoT uređaja

- U IoT ekosustavu, očekuje se da stvari surađuju jedna s drugom u svrhu ostvarivanja skupa zajedničkih ciljeva
- Povezani su standardima - Bluetooth, ZigBee, 6LoWPAN, IEEE 802.15.6 - Wi-Fi, 4G, 5G
- Svi IoT uređaji su povezani i na Internet – zbog ograničenih mogućnosti procesiranja i pohranjivanja podataka komuniciraju putem interneta sa stabilnijim računalnim platformama

- Prema IEEE, da bi uređaj bio IoT, treba imati neku senzorsku mogućnost – sposoban za praćenje jednog ili više parametara okruženja te periodičkog izvještavanja o dohvaćenim podacima
- Druga važna mogućnost prema IEEE-u je mogućnost donošenja odluka – pametni uređaji

## Ostale karakteristike

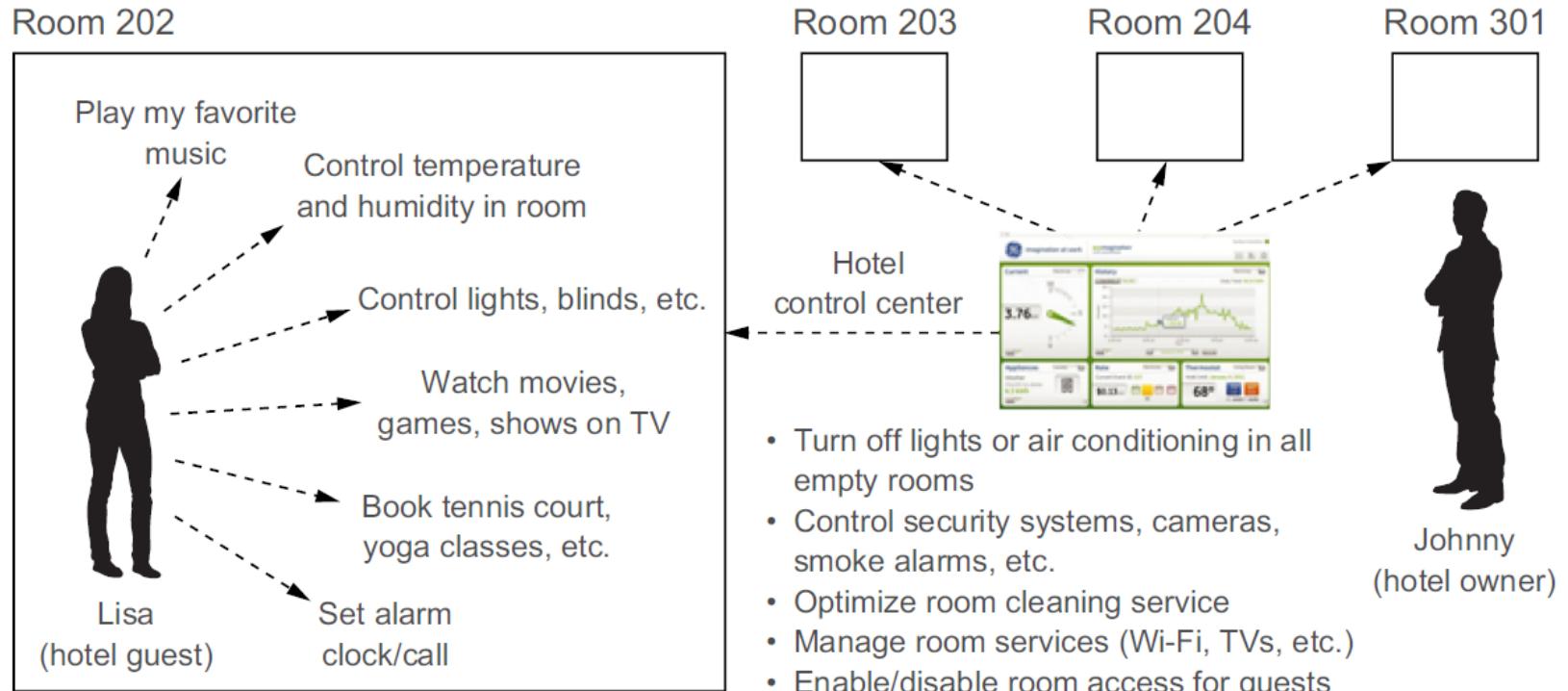
- IoT uređaji su često mobilni po prirodi (*ubiquity*) – oblak
- Samokonfiguriranje - zbog heterogenosti, IoT uređaji su često dizajnirani tako da budu samoupravljeni i samokonfigurirajući
- Programibilnost – Temeljem korisničkih zahtjeva, programeri mogu udaljeno reprogramirati uređaje

- Temeljni princip koji se slijedi za dizajn IoT middleware-a je minimiziranje ljudske intervencije i poboljšanje međumodulne interakcije tako da uređaj može raditi neprimjetno i neovisno

Svojstva IoT middleware-a:

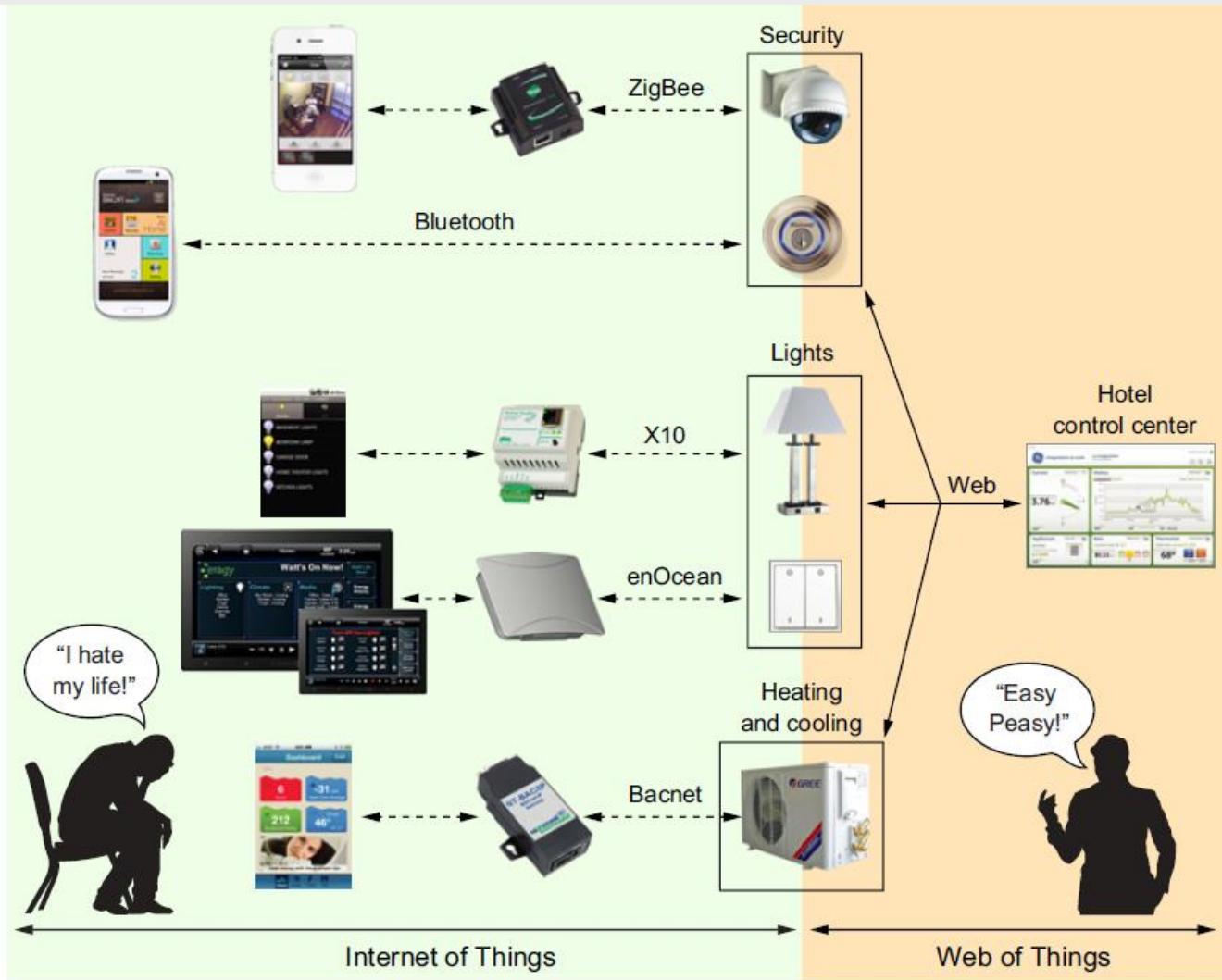
- svijest o kontekstu,
- sposobnost upravljanja resursima,
- analitička sposobnost
- prenosivost
- grafičko korisničko sučelje

# Scenarij weba stvari – povezani hotel



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

# Stotine nekompatibilnih IoT protokola



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Web stvari se bavi samo najvišim OSI slojem koji rukuje aplikacijama, uslugama i podacima.
- Rad s tako visokom razinom apstrakcije omogućuje povezivanje podataka i usluga s mnogih uređaja bez obzira na stvarne transportne protokole koje koriste.
- Nasuprot tome, IoT ne zagovara jedan protokol na razini aplikacije i obično se fokusira na niže slojevima OSI modela.
- Web stvari je mogućnost korištenja modernih web standarda na ugrađenim uređajima.

- IoT će promijeniti mnoge segmente u industrijskim, poslovnim, zdravstvenim i potrošačkim proizvodima
- Industrija će se trebati mijenjati u načinu na koji stvaraju proizvode i pružaju usluge
- Svakim danom sve više raste broj povezanih objekata
- 2021 – više od 10 milijardi aktivnih IoT uređaja
- U 2030 očekuje se oko 25 milijardi IoT uređaja
- Očekuje se da će količina podataka koju generiraju IoT uređaji doseći 73,1 ZB (zetabajta) do 2025. godine.

# Primjene IoT-a

- Zdravstvo
- Pametni grad
- Pametna kuća
- Telekomunikacije – SIM kartice
- Upravljanje nabavnim lancem

- Preventivno održavanje na novim i već postojećim tvorničkim strojevima
- Povećanje propusnosti kroz potražnju u stvarnom vremenu
- Ušteda energije
- Sigurnosni sustavi kao što su senzori topline, tlaka i curenja plina

- Pametna oprema za dom: pametno navodnjavanje, pametna garažna vrata, pametne brave, pametna svjetla, pametni termostati i pametna sigurnost.
- Nosivi uređaji: uređaji za praćenje zdravlja i kretanja, pametna odjeća/nosivi uređaji.
- Kućni ljubimci: sustavi za lociranje kućnih ljubimaca, pametna vrata za pse.

# Upotreba IoT-a u maloprodaji

- Ciljano oglašavanje
- Lociranje poznatih ili potencijalnih kupaca
- Mjerenje rizika osiguranja vozača
- Digitalno oglašavanje u maloprodaji i ugostiteljstvu
- Beaconing sustavi unutar zabavnih sadržaja, konferencija, koncerata, zabavni parkova i muzeja.

- Kućna njega pacijenata
- Učenje modela prediktivne i preventivne zdravstvene zaštite
- Demencija i njega i praćenje starijih osoba
- Praćenje bolničke opreme i opskrbe
- Farmaceutsko praćenje i sigurnost
- Medicina na daljinu
- Istraživanje lijekova
- Indikatori pada pacijenata

- Praćenje flote i svijest o lokaciji
- Identifikacija i praćenje vagona
- Praćenje imovine i paketa unutar flote
- Preventivno održavanje vozila na cesti

- Pametne tehnike navodnjavanja i gnojidbe za poboljšanje prinosa
- Pametna rasvjeta u gniježđenju ili uzgoju peradi za poboljšanje prinosa
- Praćenje zdravlja stoke i imovine
- Preventivno održavanje opreme za udaljenu poljoprivrodu
- Istraživanja zemljišta temeljena na dronovima
- Učinkovitost lanca opskrbe od farme do tržišta uz praćenje imovine
- Robotska poljoprivreda
- Praćenje vulkanskih i rasjednih linija za predviđanje katastrofa

- Kontrola onečišćenja i regulatorna analiza putem senzora okoliša
- Povećanje učinkovitosti i poboljšanje troškova kroz uslugu gospodarenja otpadom
- Poboljšan protok prometa i uštede goriva putem pametne kontrole semafora
- Pametne kamere za praćenje kriminala
- Pametna parkirališta za automatsko pronalaženje najboljeg parkirnog mjesta na zahtjev



- Sudip Misra, Subhadeep Sarkar, Subarna Chatterjee: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things, CRC Press, 2019
- Dominique Guinard, Vlad Trifa: Building the Web of Things: With examples in Node.js and Raspberry Pi, Manning Publications, 2016
- Perry Lea: Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure, edge computing, analytics, and security, Packt Publishing, 2018

# Računalstvo u oblacima

Darko Andročec

- Računalstvo u oblaku (cloud computing) je paradigma informacijske tehnologije koja osigurava isporuku usluga u smislu softvera, platforme i infrastrukture na zahtjev.
- Definicija prema NIST-u:

„Računalstvo u oblaku je model za omogućavanje sveprisutnog, praktičnog pristupa mreži na zahtjev zajedničkom skupu računalnih resursa koji se mogu konfigurirati (npr. mreže, poslužitelji, pohrana, aplikacije i usluge) i brzo osigurati te pokrenuti uz minimalan napor upravljanja ili interakciju s pružateljem usluga.“

- Javni (public)
- Privatni (private)
- Hibridni (hybrid)
- Zajednički oblak (community cloud)

- Cjelokupna infrastruktura oblaka dostupna je korisnicima javno preko interneta
- Primjeri: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform
- Krajnji korisnici mogu dobiti usluge VM-a, pohrane u oblaku ili izvršavanja aplikacija; usluge mogu biti besplatne ili plativo na temelju korištenja

## Prednosti:

- Ublažava zahtjeve ulaganja i periodičnog održavanja on-premise IT resursa
- Manje je rasipanja resursa jer kupci jednostavno mogu platiti samo resurse koje stvarno potroše

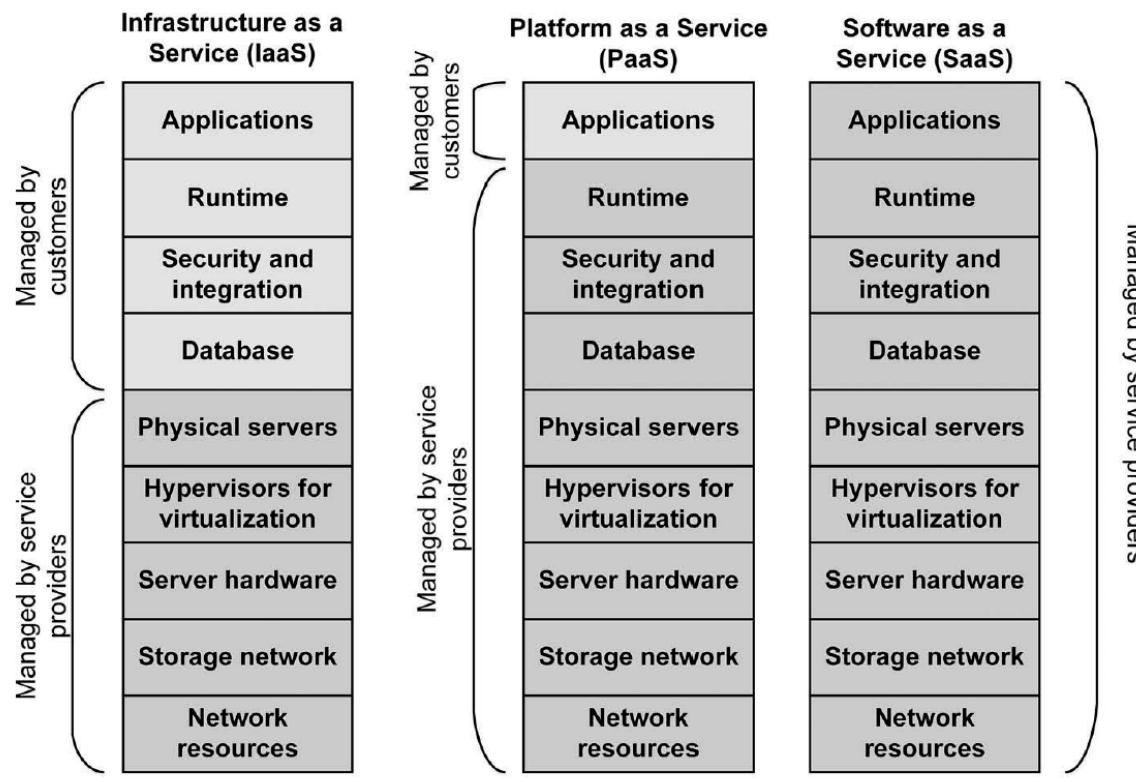
- Infrastruktura se nalazi i održava u samoj organizaciji, ali je arhitektura zadana
- Primjeri: Amazon Elastic Compute Cloud (EC2), IBM's Blue Cloud, Sun Cloud, Google App Engine, Windows Azure Services Platform
- Ispunjava zahtjeve usluga i aplikacija unutar organizacije i nije za javnost.
- Prednosti:
- Ima sve prednosti javnih cloud infrastruktura.
- Tvrta može zadržati punu kontrolu nad resursima i hardverom poslužitelja u oblaku.

## Hibridni oblak

- Kombinacija javnog i privatnog oblaka
- Na primjer, organizacija može koristiti javni oblak za pohranu i obradu podataka te koristiti privatni oblak za postavljanje i izvršavanje starijih aplikacija.
- Ako se zahtjevi tvrtkinih usluga u oblaku obično povećavaju u određeno doba godine, može se odabrat da se veći promet rješava koristeći hibridnu strategiju.
- Međutim, rješenja hibridnog oblaka mogu biti skupa, pa ih treba koristiti mudro i strateški.

- model usluge u oblaku koji pruža rješenje računalstva u oblaku ograničenom broju pojedinaca ili organizacija kojima zajednički upravljaju sve organizacije koje sudjeluju ili pružatelj usluga kojim upravlja treća strana

- Softver kao servis – SaaS
- Platforma kao servis – PaaS
- Infrastruktura kao servis – IaaS



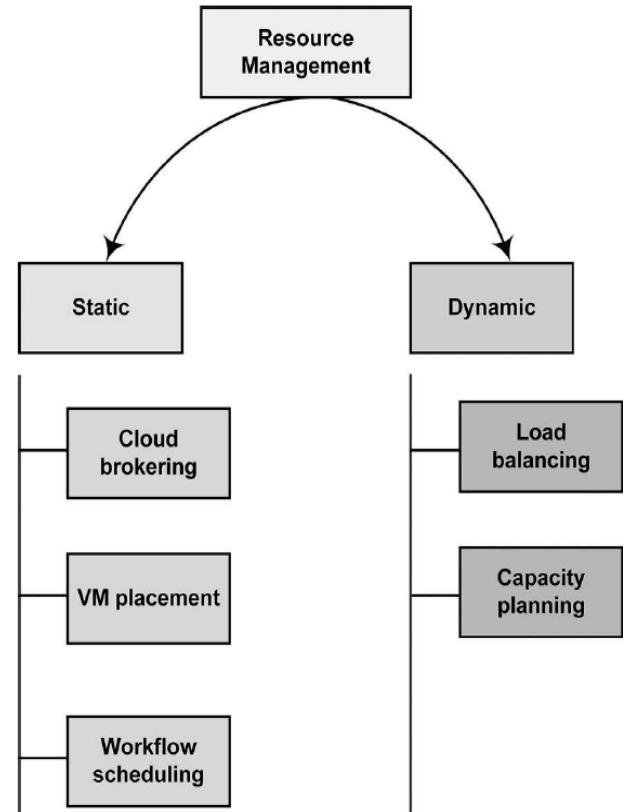
- Pružatelji nude softver kao servis u smislu cjelovitog proizvoda
- Krajnji korisnici ne bave se mehanizmima obrade i pružanja usluge, te ne trebaju instalirati ili održavati softver
- Kako bi korisnici oblaka mogli pristupiti i koristiti bilo koji softver preko Interneta bez brige o specifikacijama, konfiguraciji, ili internim pojedinostima softvera
- Google Workspace, Cisco WebEx

- Općenito ciljaju na programere koji namjeravaju graditi aplikacije na određenim platformama.
- PaaS omogućuje jednostavno, brzo i praktično upravljanje aplikacija od strane programera.
- Windows Azure, Google App Engine, and Apache Stratos

- Prvenstveno se koristi za upravljanje aplikacijama.
- Usluge IaaS oblaka uključuju pristup i korištenje pohrane, virtualizacije, hardvera i mrežnih resursa
- Neki uobičajeni primjeri IaaS-a su AWS, Microsoft Azure, Alibaba, GCP

# Upravljanje resursima u oblaku

- Statičko i dinamičko upravljanje resursima
- U statičkoj dodjeli resursa, zahtjevi za resursima su poznati a priori, dok za dinamičku dodjelu resursa zahtjevi stižu dinamički tijekom izvođenja aplikacije
- Upravljanje resursima u oblaku je vrlo složen proces, jer ovisi o ponašanju korisnika, obrascima radnog opterećenja i opterećenju sustava.



- IaaS oblak pruža maksimalnu fleksibilnost usluga, budući da programeri mogu dizajnirati, dodijeliti i isprobavati vlastite virtualne strojeve
- Korisnicima se naplaćuju samo zakupljeni resursi i vrijeme trajanja korištenja resursa
- Zbog velikog broja korisnika, performanse VM-a je kritičan problem

- Proučavanje i praksa projektiranja, proizvodnje, korištenja i odlaganja računala, poslužitelja i povezanih podsustava – kao što su monitori, pisači, uređaji za pohranu i umrežavanje, te komunikacijski sustavi – učinkovito i djelotvorno s minimalnim utjecajem ili bez utjecaja na okoliš
- Ovaj aspekt ima povećan značaj u domeni računalstva u oblaku, budući da su obrade u oblaku složene zbog korištenog softvera, poslužene aplikacije, potrošenih resursa i pruženih usluga
- Zeleno računalstvo na platformama u oblaku bavi se energetski učinkovitim modeliranjem performansi i optimizacijom podatkovnih centara

- Pohrana podataka
- Društveno umreživanje
- Zdravstvo
- Državne institucije
- Vojska
- Naplata
- Bankarstvo
- Kupnja
- Online edukacija i učenje

- Čvorovi senzora – prvo žično povezani, danas uglavnom bežično
- Bežični senzori su postali popularniji zbog njihove prenosivosti i jednostavnosti montaže
- U smislu sposobnosti procesiranja i pohrane, senzorski se čvorovi smatraju ograničeni resursima
- Prvi senzor – senzor temperature 1860 – Wilhelm von Siemens
- Senzor - uređaj koji emitira električni signal kao odgovor na mjerljivu veličinu

- Prve mreže senzora koristile su se za vojne svrhe – npr. SOSUS – strateški raspoređeni akustički senzori za praćenje sovjetskih podmornica
- 21. st. bežični senzori sa sposobnošću manjeg računalnog procesiranja
- Pojavilo se nekoliko komunikacijskih protokola: Bluetooth, ZigBee i WiMax

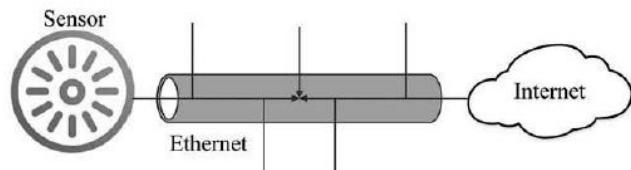
# Karakteristike konvencionalnih mreža bezičnih senzora (WSN) (1)

- Hardver - četiri osnovna modula: senzorski i aktivacijski modul, komunikacijski modul, modul za obradu i modul napajanja – moduli imaju open-source operacijski sustav
- Komunikacija – svaki čvor posjeduje primopredajnik koji funkcionira unutar određenog raspona - IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (ZigBee), IEEE 802.16 (WiMax)
- Usmjeravanje - čvorovi općenito usmjeravaju pakete podataka višestrukom rutom – sheme usmjeravanja mogu biti: usmjerene na podatke, hijerarhijske, bazirane na lokaciji ili bazirane na kvaliteti usluge

## Karakteristike konvencionalnih mreža bezičnih senzora (WSN) (2)

- Interni softver – open-source operacijski sustavi – npr. TinyOS
- Rad s podacima – obrada podataka, kalibracija, fuzija, agregacija, odlučivanje – dinamička obrada upita i kanaliziranje upita kroz mrežu
- Upravljanje mrežom - pojedinosti o mrežnoj arhitekturi i komunikaciji, standardi, tolerancija grešaka mreže, učinkovitost resursa i komunikacijske frekvencije

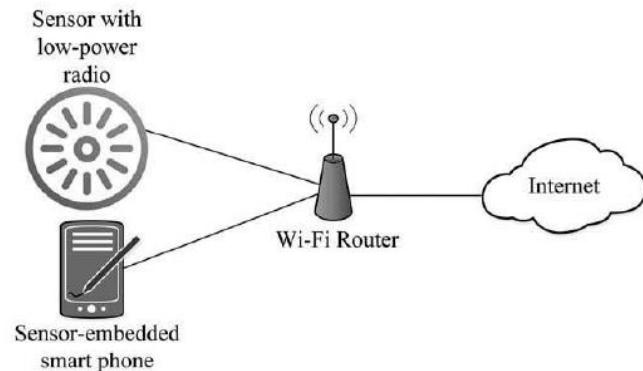
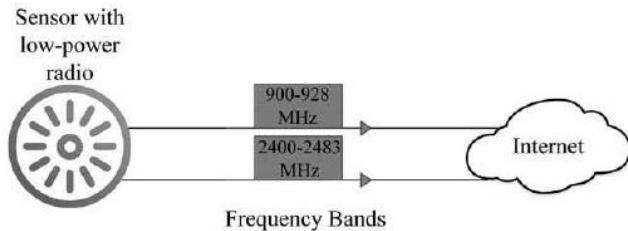
- Integracija WSN-a na Internet



(a) Before 1980s



(b) Between 1980s and 1990s



# Tipovi senzora

- Infracrveni senzori
- Senzori vida – opremljeni kamerom
- Senzori optičkih vlakana
- Senzori razine
- Senzori protoka - mjere brzinu protoka tekućina
- Senzori plinova
- HVAC senzori – grijanje, ventilacija i klima
- Senzori pokreta
- Senzori dodira ekrana
- Senzori praćenja okruženja – temperatura zraka, vlažnost, količina oborina, atmosferski tlak

- Senzorski i aktuacijski modul, komunikacijski modul, procesorski modul, modul za napajanje, moduli specifični za pojedinu primjenu
- Hardver za obradu signala
- Pametni senzori – opremljeni inteligencijom za upravljanje izvođenjem i donošenje odluka
- Prednosti pametnog senzora uključuju bolje održavanje, manje zastoja, veća pouzdanost, otpornost na pogreške, manja težina i bolja arhitektura sustava.

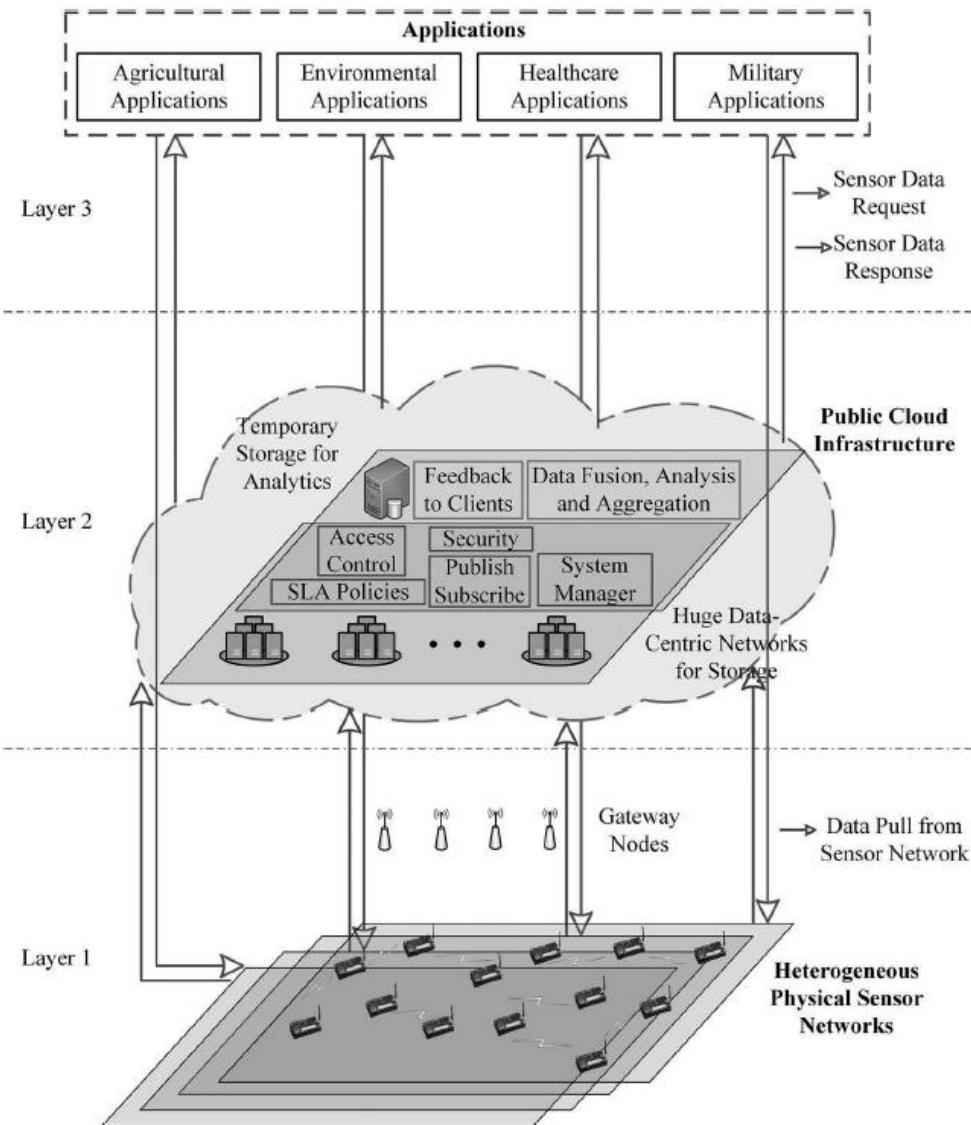
- Vojne primjene
- Praćenje cilja – nadzor i otkrivanje uljeza
- Zdravstvo
- Kućanstva
- Očuvanje okoliša
- Praćenje i kontrola vozila
- Praćenje prirodnih katastrofa – potresi, vulkanske erupcije

# Izazovi i ograničenja

- Postavljanje WSN-a
- Topologija
- Energija
- Održavanje
- Hardver
- Diseminacija

- WSN imaju mnoga ograničenja
- Integracija WSN-a i oblaka bitna je zbog malih spremišta podataka u WSN-u, ograničenih procesnih mogućnosti i mogućnosti analize povijesnih podataka

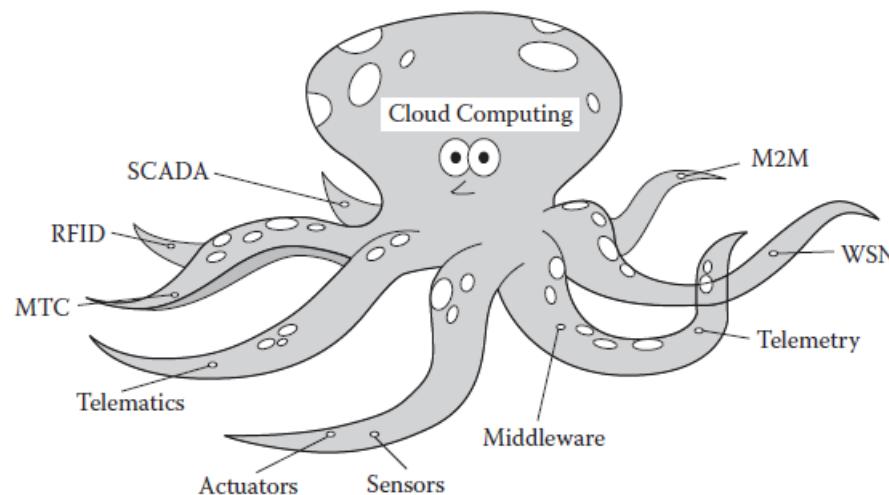
# Integracija WSN-a i oblaka



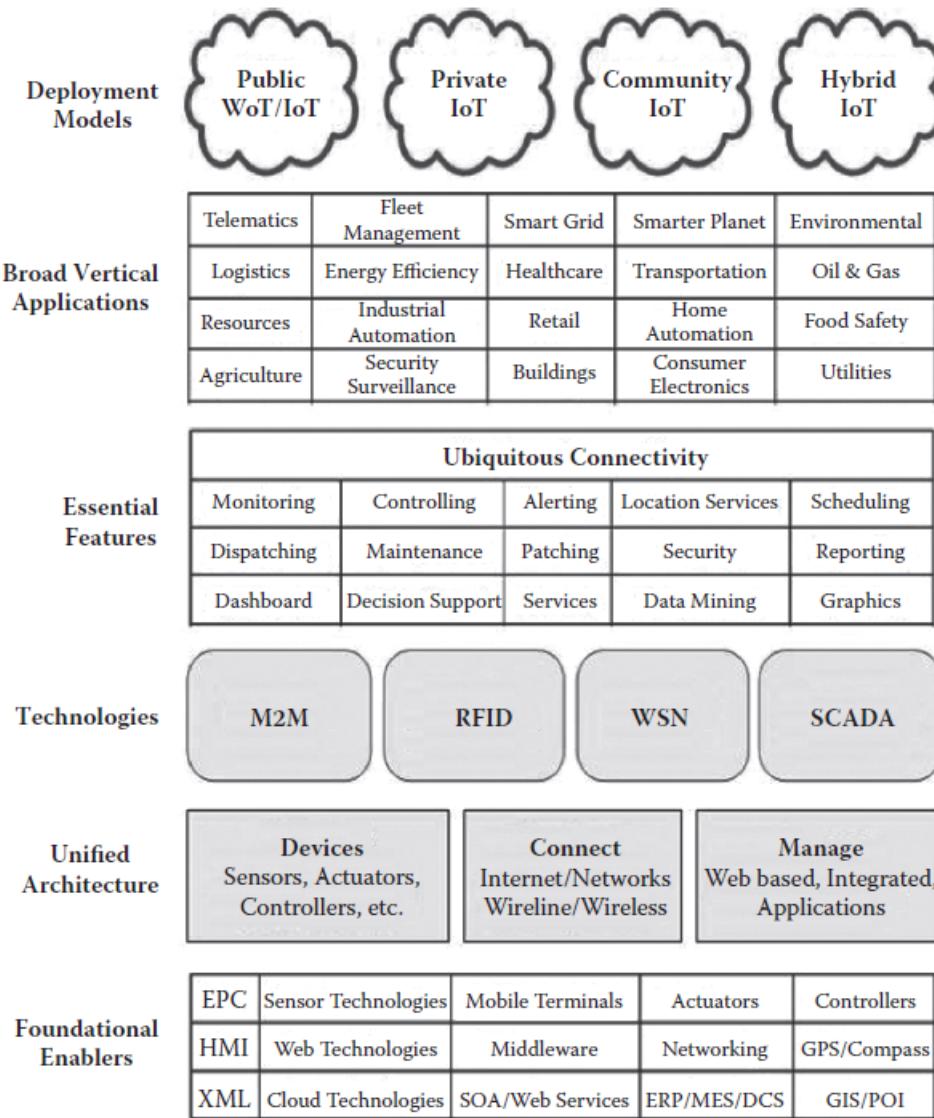
## Primjeri integracije

- Poljoprivreda - Računalstvo u oblaku dobro se integrira s poljoprivrednim senzorskim mrežama za pružanje usluga poljoprivrednicima po vrlo niskoj cijeni.
- Zdravstvo – senzori na tijelu pacijenata
- Ekologija – praćenje prašuma – podaci za dugoročno korištenje u oblaku, za praćenje globalnog zatopljenja

- Mobilno računarstvo, računalstvo u oblacima i IoT su međusobno isprepleteni
- Računalstvo u oblaku je tehnologija koja omogućuje IoT
- Oblak i IoT najbolje je promatrati kao kontinuum internetske povezanosti s oblakom kao "glava" i IoT kao "rep" hobotnice kao što je prikazano na slici



# Arhitektura oblaka stvari



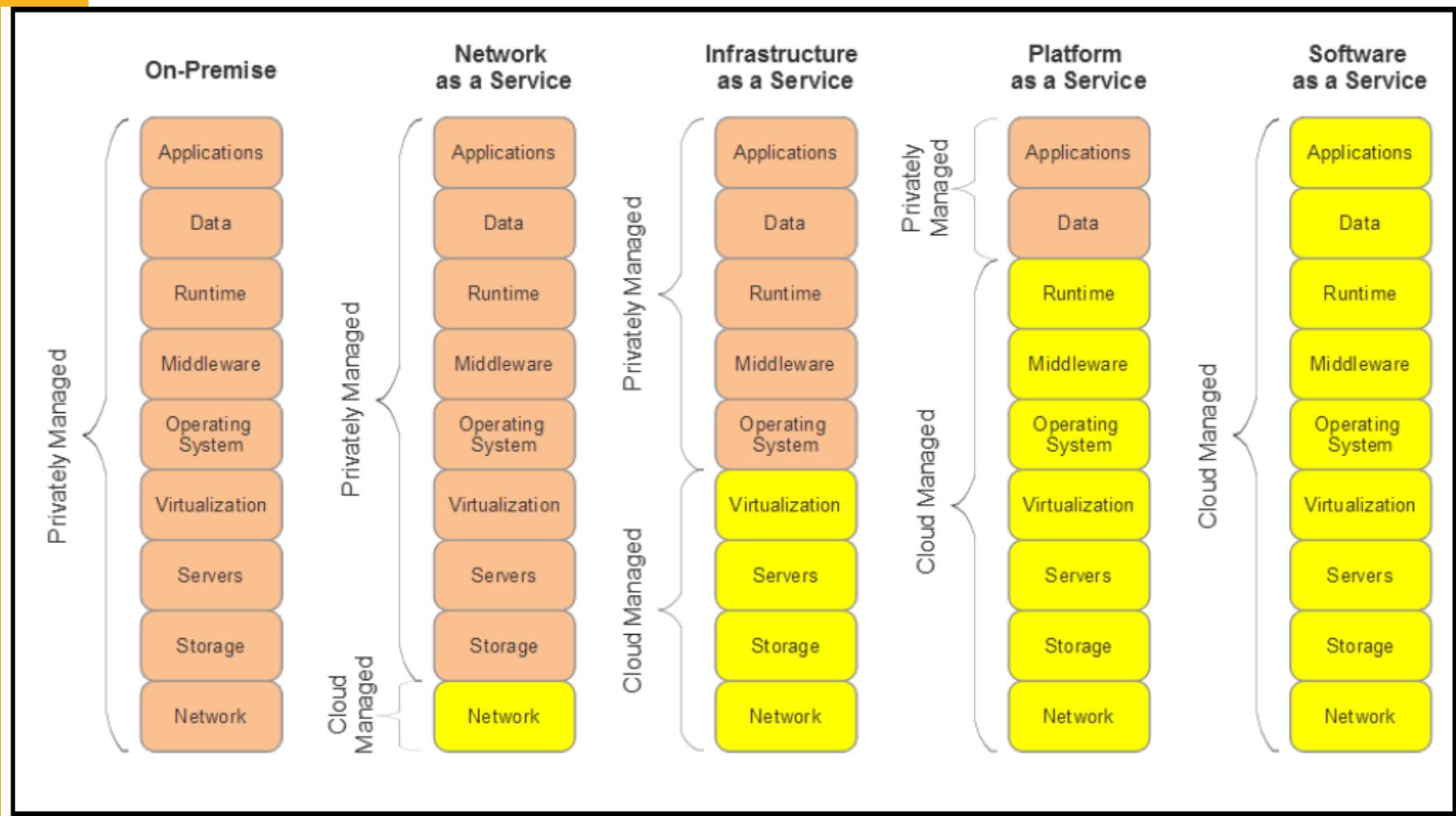


# Topologija oblaka i fog-a

Darko Andročec

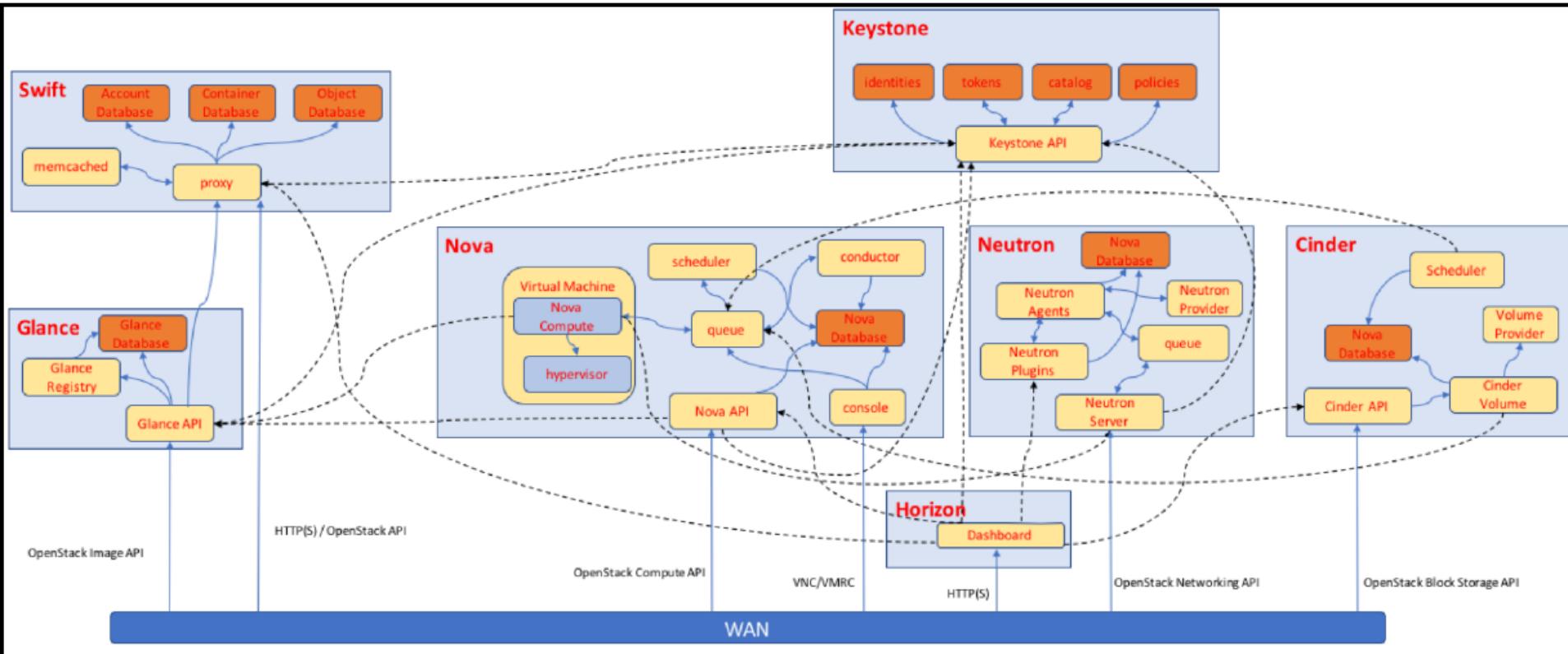
- Bez računalstva u oblacima, ne bi postojao tako veliki tržišni rast IoT-a
- Vrijednost IoT-a je u podacima koje IoT stvara, a ne u velikoj količini pametnih uređaja
- Oblak omogućuje da se IoT uređaji spoje i da se mogu agregirati podaci
- Oblaci su obično veliki podatkovni centri koji pružaju usluge kupcima po modelu plaćanja za korištenje.
- Ovo korisniku daje osjećaj neovisnosti o lokaciji.
- Resursi su elastični (što znači skalabilni), a usluge se plaćaju za korištenje, generirajući stalni tok prihoda za davatelja usluga.

# Arhitekturalni modeli oblaka



- OpenStack je Apache 2.0 licencirani okvir otvorenog koda koji se koristi za izgradnju platformi u oblaku.
- To je prvenstveno IaaS i u upotrebi je od 2010.
- Arhitektura ima sve glavne komponente drugih sustava u oblaku, uključujući računalnu obradu i uravnoteženje opterećenja; komponente za pohranu, uključujući sigurnosno kopiranje i oporavak; komponente za umrežavanje, nadzorne ploče, sigurnost i identitet, podaci i analitički paketi, implementacija alata, monitora i mjerača te aplikacijskih usluga.

# Arhitektura OpenStacka



Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure

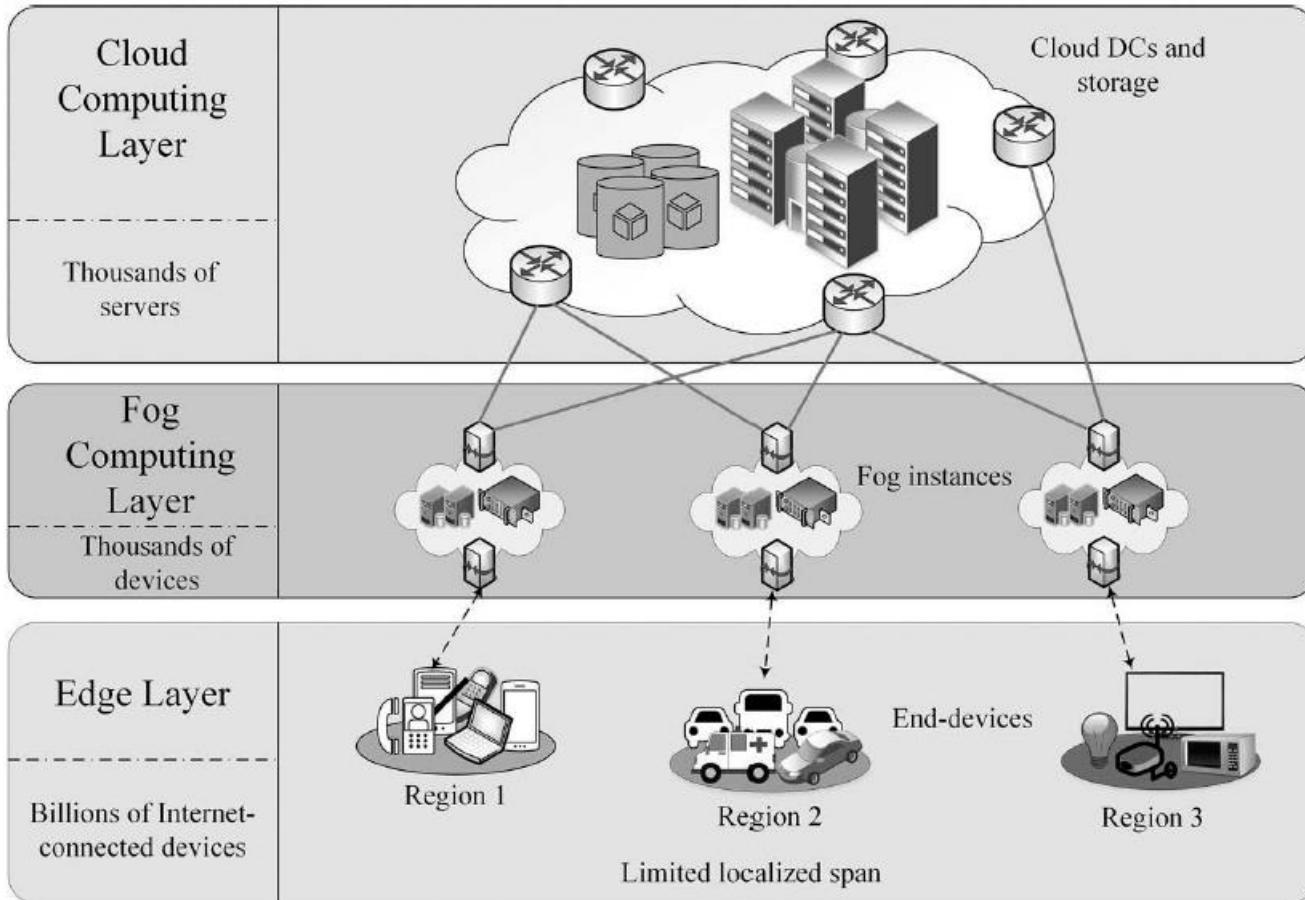
- Uređaji možda nisu IP-kompatibilni – npr. Bluetooth Low Energy (BLE) i Zigbee – potreban je edge gateway za prevođenje
- Efekti kašnjenja u oblaku - odgovor u stvarnom vremenu kritičan je u mnogim IoT aplikacijama i prisiljava da se procesiranje približi krajnjem uređaju.
- Latencija oblaka bit će reda desetaka ako ne i stotina milisekundi bez obračunavanja svih troškova obrade dolaznih podataka

- Rubno računalstvo - premještanje obrade blizu mesta gdje se nalaze i generiraju podaci.
- U slučaju IoT-a, rubni uređaj mogao bi biti sam senzor s malim mikrokontrolerom ili ugrađeni sustav sposoban za WAN komunikaciju.
- Nekad će rub (edge) biti pristupnik (gateway) u arhitekturama s posebno ograničenim krajnjim točkama spojenih na pristupnik.
- Obrada rubova također se obično spominje u kontekstu obrade od stroja do stroja gdje postoji uska korelacija između ruba (klijenta) i poslužitelja smještenog negdje drugdje.

- Evolucijsko proširenje računalstva u oblaku na rub
- Dijeli okvirni API i komunikacijski standard s ostalim fog čvorovima i/ili servisima u oblaku.
- Čvorovi magle proširenja su oblaka, dok rubni uređaji mogu, ali i ne moraju uključivati oblak.
- Osim toga, fog može biti u više razina hijerarhije
- Fog čvorovi su slični infrastrukturi hibridnog oblaka, samo su manje moćni što se tiče obrade i pohrane
- Fog Computing je visoko virtualizirana platforma koja pruža računanje, pohranu i mrežne usluge između krajnjih uređaja i tradicionalni podatkovnih centara za računalstvo u oblaku, koje se obično, ali ne isključivo nalazi na rubu mreže.

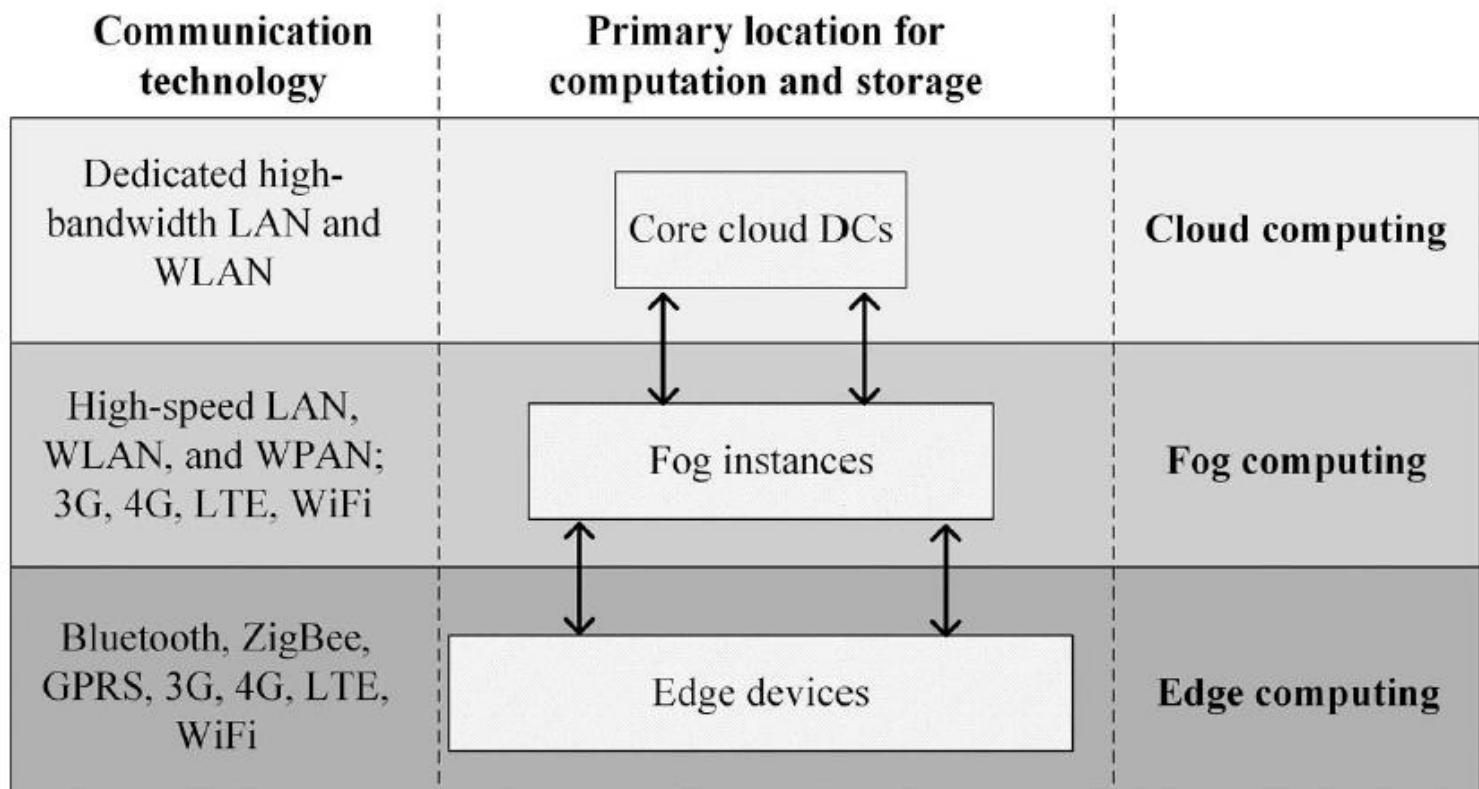
- Izraz „magla“ je osmisnila zaposlenica Cisca Ginny Nichols sredinom 2012. kao kraticu od "From Core to edge.."
- Pojam magle bio je predložen kao metafora – kao što se magla u prirodi stvara blizu tla, računalstvo u magli ukazuje na obradu bliže rubu mreže.
- Pojam također služi kao savršena suprotnost za računalstvo u oblaku, koje se odvija daleko od mrežnog ruba.

# Paradigma računalstva u magli



Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

# Usporedba različitih računalnih paradigm

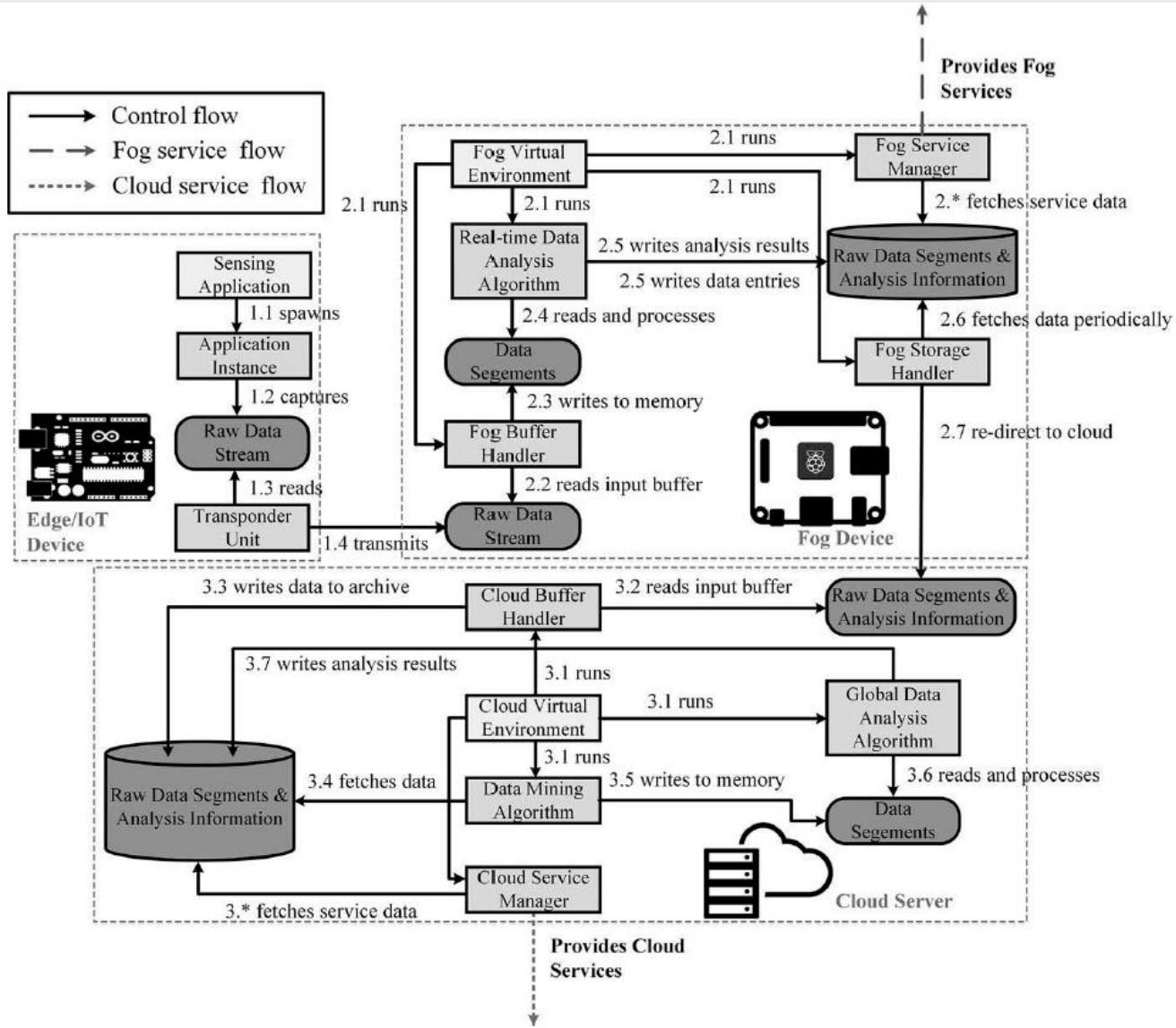


# Karakteristike računalstva u magli

- Blizina krajnjim korisnicima
- Mala latencija usluge
- Usluga prepoznavanja lokacije
- Gusta distribucija
- Različitost fog uređaja
- Podrška mobilnosti uređaja
- Dinamičko skaliranje
- Dominira bežični pristup
- Interoperabilnost

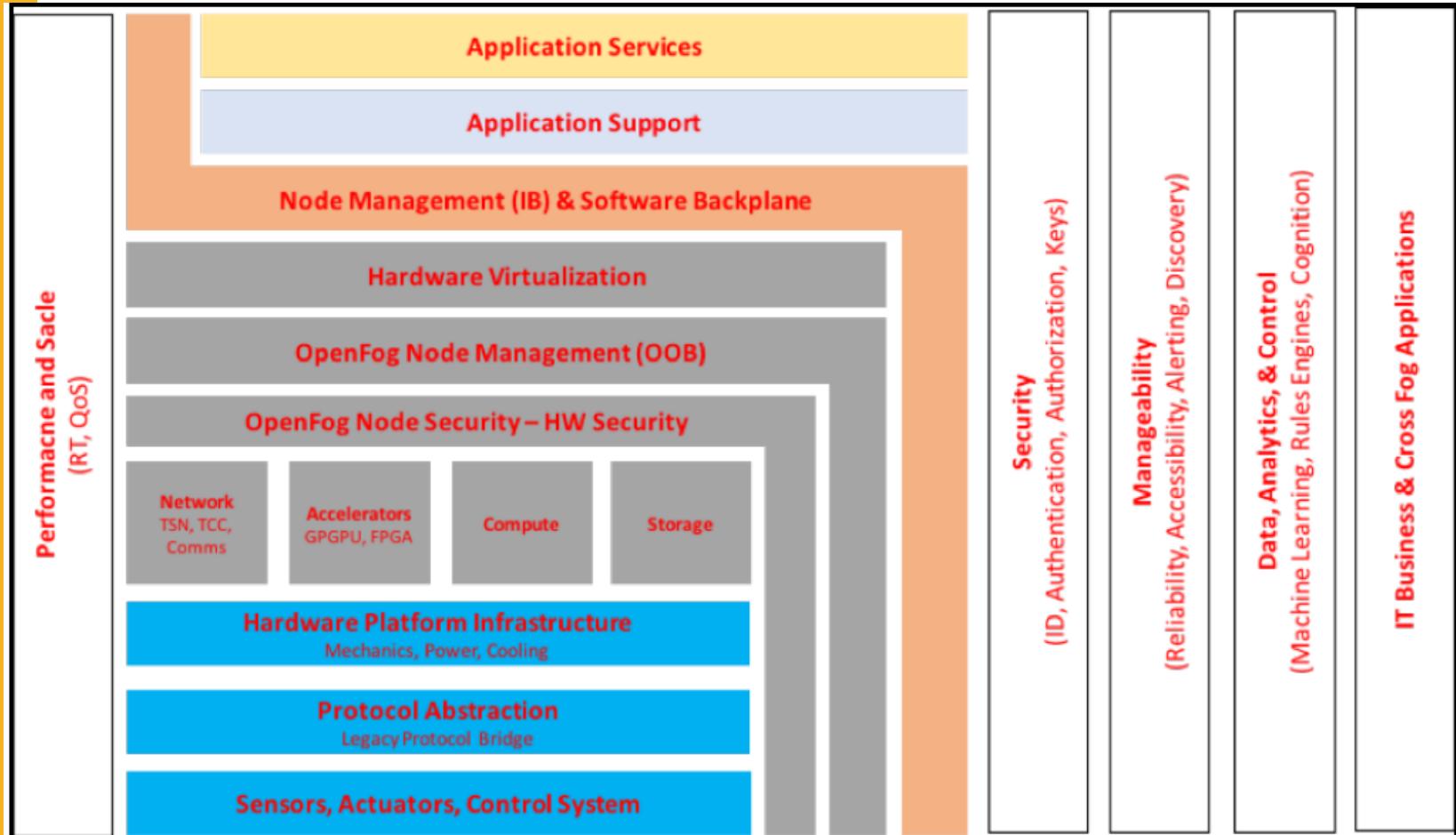
- Poboljšana kvaliteta usluge (QoS)
- Optimizacija troška – smanjena količina migracije podataka
- Ograničavanje uskog grla - prednost u odnosu na klasični model računalstva u oblaku ograničavanjem mrežnog uskog grla
- Virtualizacija specifična lokaciji – svi fog zahtjevi se prvo obrađuju u srednjem sloju fog-a, pa se daje prioritet zahtjevima koji su osjetljivi na latenciju usluge
- Računalstvo u magli ima najbolje rezultate u kontekstu IoT-a i budućeg interneta, gdje se nalazi ogroman broj uređaja kojima su potrebne usluge specifične za određenu lokaciju u stvarnom vremenu

# Model fog servisa



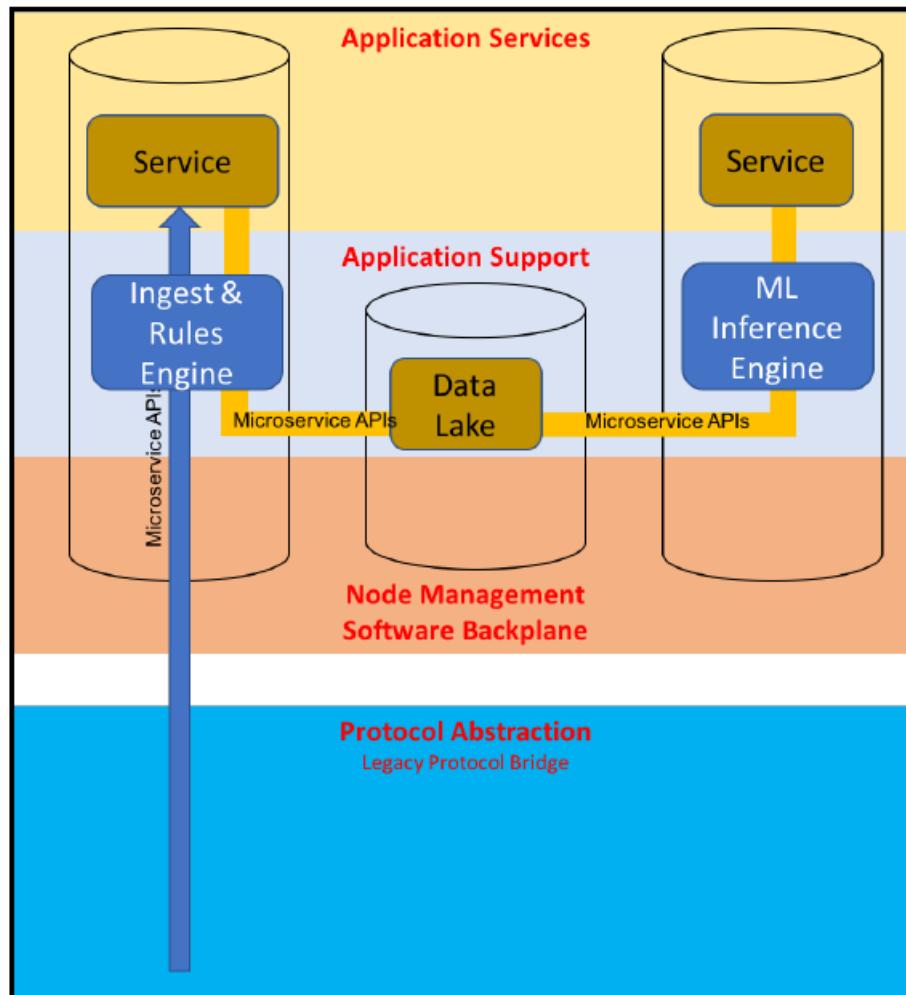
- Zdravstvo – sveprisutno praćenje zdravlja i e-zdravstvo – automatizirano i udaljeno praćenje fiziologije
- Upravljanje pametnim vozilima – autonomna vozila, pametni semafori, pametni prijevozni sustavi
- Pametni gradovi – pametne građevine, pametno upravljanje i distribucija energijom, pametna poljoprivreda, pametno upravljanje otpadom
- Pametno upravljanje podacima – IoT – podaci se formiraju u formi struje velikom brzinom (streams at a high velocity)

# OpenFog referentna arhitektura



Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure

# OpenFog: aplikacijski servisi



- Upravljanje aplikacijom (identifikacija slike virtualnog stroja, provjera slike, implementacija i autentifikacija)
- Alati za logiranje
- Registracija komponenti i usluga
- Runtime „motori“ (spremniči, VM)
- Runtime jezici (Node.js, Java, Python)
- Aplikacijski poslužitelji (Tomcat)
- Sabirnice poruka (RabbitMQ)
- Baze podataka i arhivi (SQL, NoSQL, Cassandra)
- Analitički okviri (Spark)
- Web poslužitelji (Apache)
- Alati za analitiku (Spark, Drool)

- Otkrivanje usluge: Omogućuje ad hoc modele povjerenja od foga do foga.
- Otkrivanje čvorova: Omogućuje dodavanje maglovitih čvorova i spajanje u klaster slično tehnikama klasteriranja u oblaku.
- Upravljanje stanjem: Omogućuje različite računalne modele za stateful i stateless obrade s više čvorova.
- Upravljanje Pub/Sub: Omogućuje guranje podataka umjesto povlačenja. Osim toga, dopušta razine apstrakcije u izradi softvera.

# OpenFog: sigurnost čvora

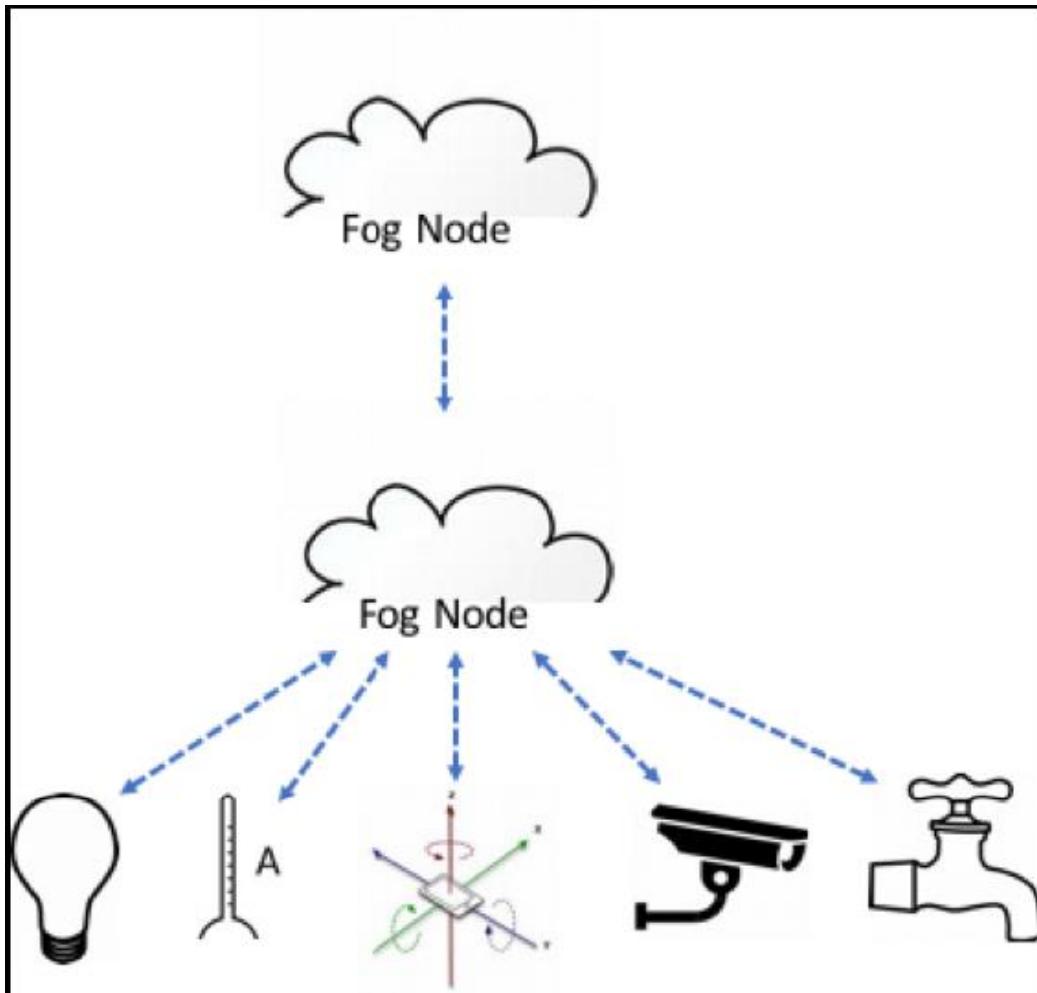
- Enkripcija
- Nadzor neovlaštenog otvaranja i fizičke sigurnosti
- Inspekcija i praćenje paketa

- Mreža – uloga fizičkog usmjeravanja prema drugim čvorovima
- Akceleratori – GPU, FPGA
- Računalna obrada – izvršavanje zadataka, praćenje i pružanje resursa, balansiranje opterećenja
- Pohrana – RAM, diskovi, flash, RAID, enkripcija podataka

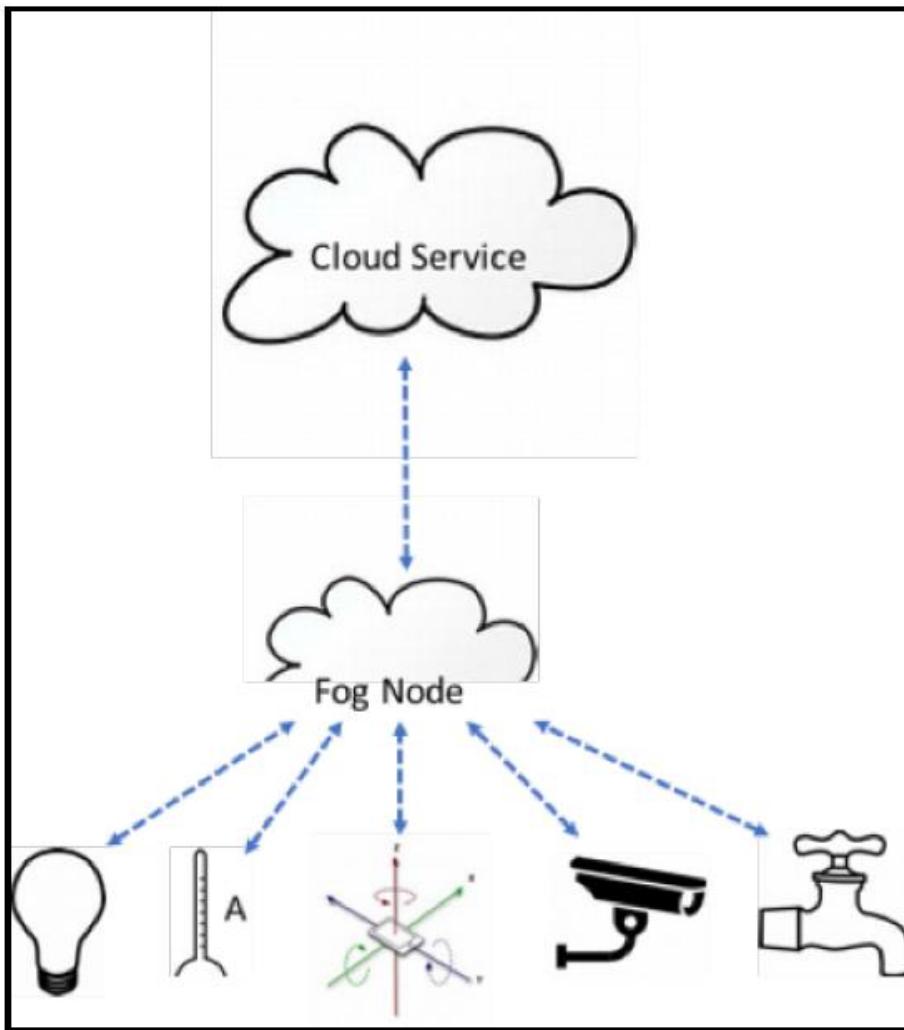
- Ekstenzija AWS-a koja omogućuje programerima skidanje klijenta za fog, gateway ili pametnu stvar
- AWS IoT Greengrass okolina za izvođenje na rubu i usluga u oblaku otvorenog koda za izgradnju, implementaciju i upravljanje softverom uređaja.
- Fog okvir
- Namjera je pružiti rješenje za smanjenje latencije i vremena odziva, ponovno korištenje troškova propusnosti i pružanje sigurnosti za rub

- Topologije magle (fog) mogu postojati u mnogim oblicima, a arhitekt treba uzeti u obzir nekoliko aspekata prilikom projektiranja end-to-end sustava magle.
- Fog mreža može biti jednostavna poput spajanja rubnog usmjerivača s omogućenom maglom senzora u uslugu u oblaku.
- Također može narasti u složenosti do višeslojne hijerarhije magle s različiti stupnjevi sposobnosti obrade i uloge na svakoj razini istovremeno distribuiraju te obradom opterećenja kada i gdje je to potrebno

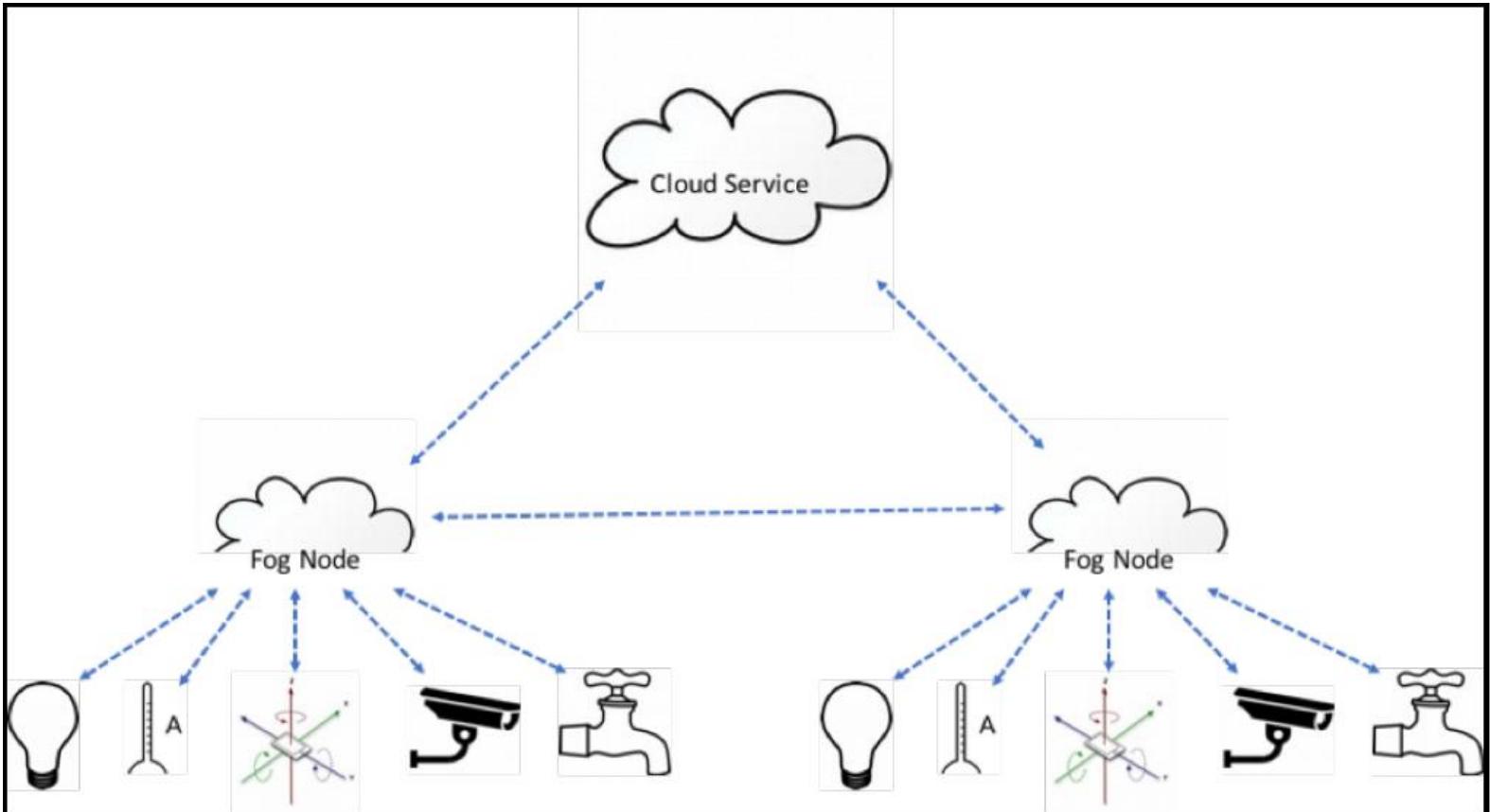
# Jednostavna topologija foga



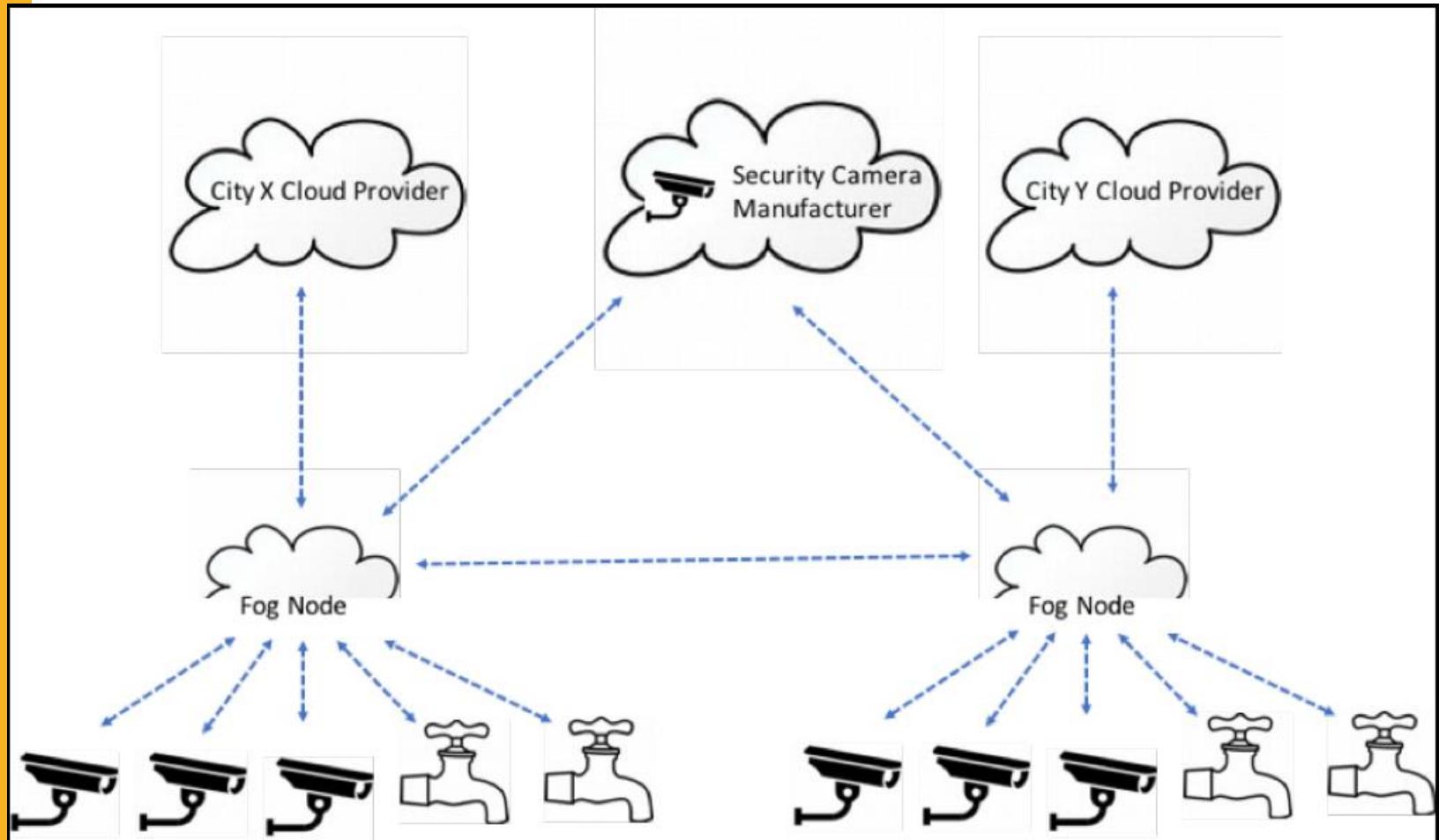
# Fog topologija s oblakom



# Višestruki fog čvorovi

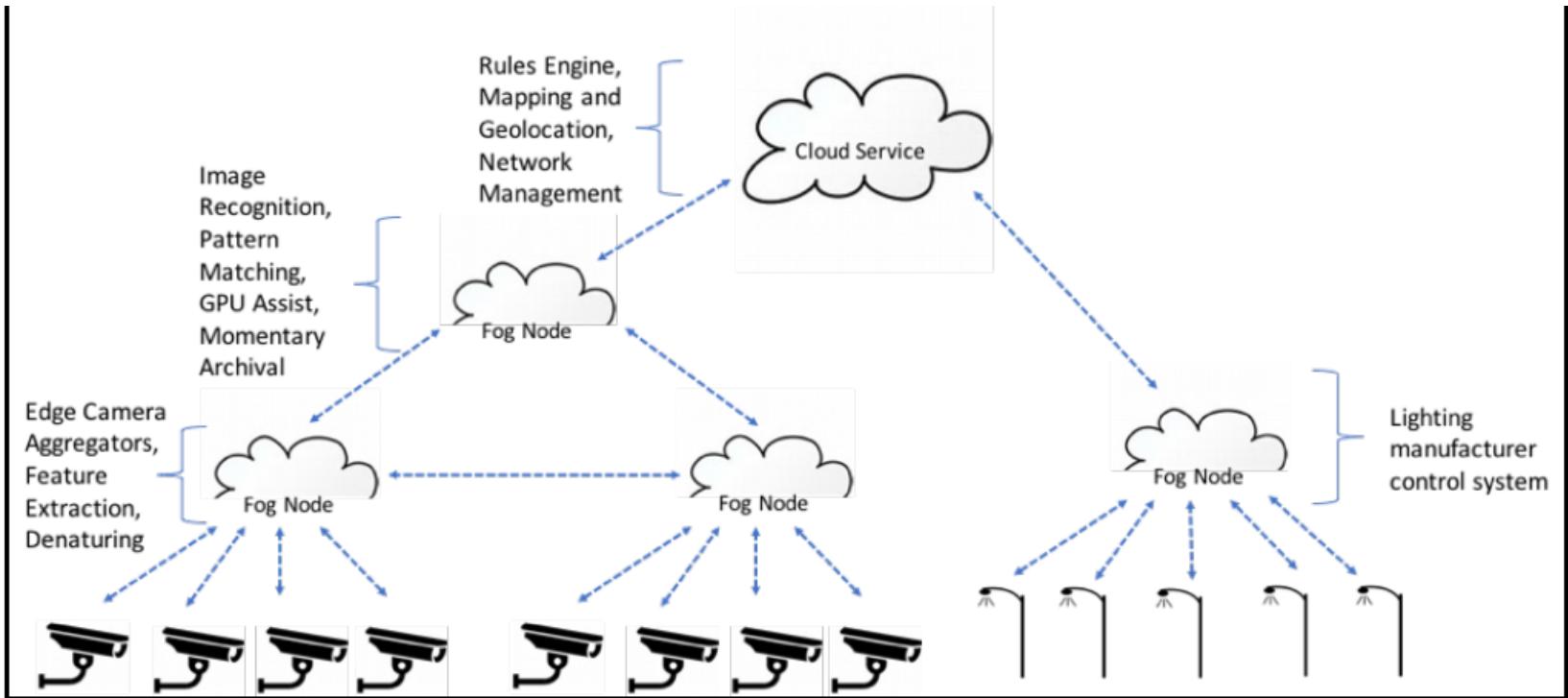


# Više pružatelja oblaka i više fog čvorova



Internet of Things for Architects: Architecting IoT solutions by implementing sensors, communication infrastructure

# Višeslojna topologija foga



- Cilj IoT-a je analiziranje podataka i donošenje smislenih zaključaka iz podataka senzora
- Kada skaliramo na tisuće do milijune i potencijalno milijarde objekata koji komuniciraju i neprekidno stvaraju podatke, moramo uvesti napredne alate za unos, pohranu, analiziranje i predviđanje značenja iz ovog mora podataka.
- Tu nam pomažu oblak i fog



# Middleware i IoT

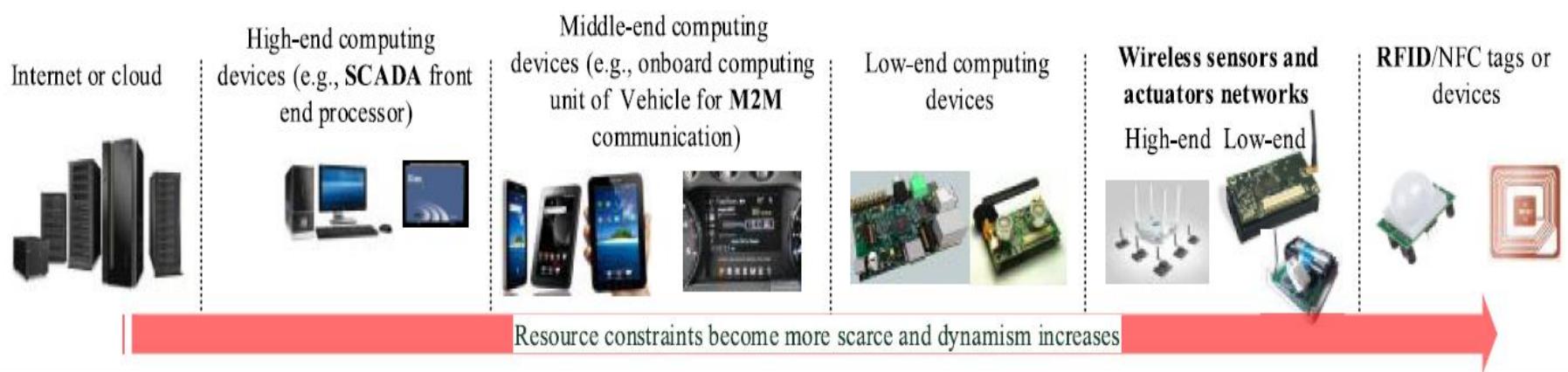
Darko Andročec

# Osnovni pojmovi (1)

- M2M – koristi uređaje (npr. uređaj u vozilu) da se primijete događaji (poput kvara motora) putem mrežne veze sa središnjim poslužiteljem – događaje pretvara u važne informacije (npr. greška upozorenja što treba popraviti na motoru).
- RFID - koristi radio valove za prijenos podataka s elektroničkog uređaja oznaka pričvršćena na objekt na središnji sustav kroz čitač u svrhu identifikacije i praćenja objekta.

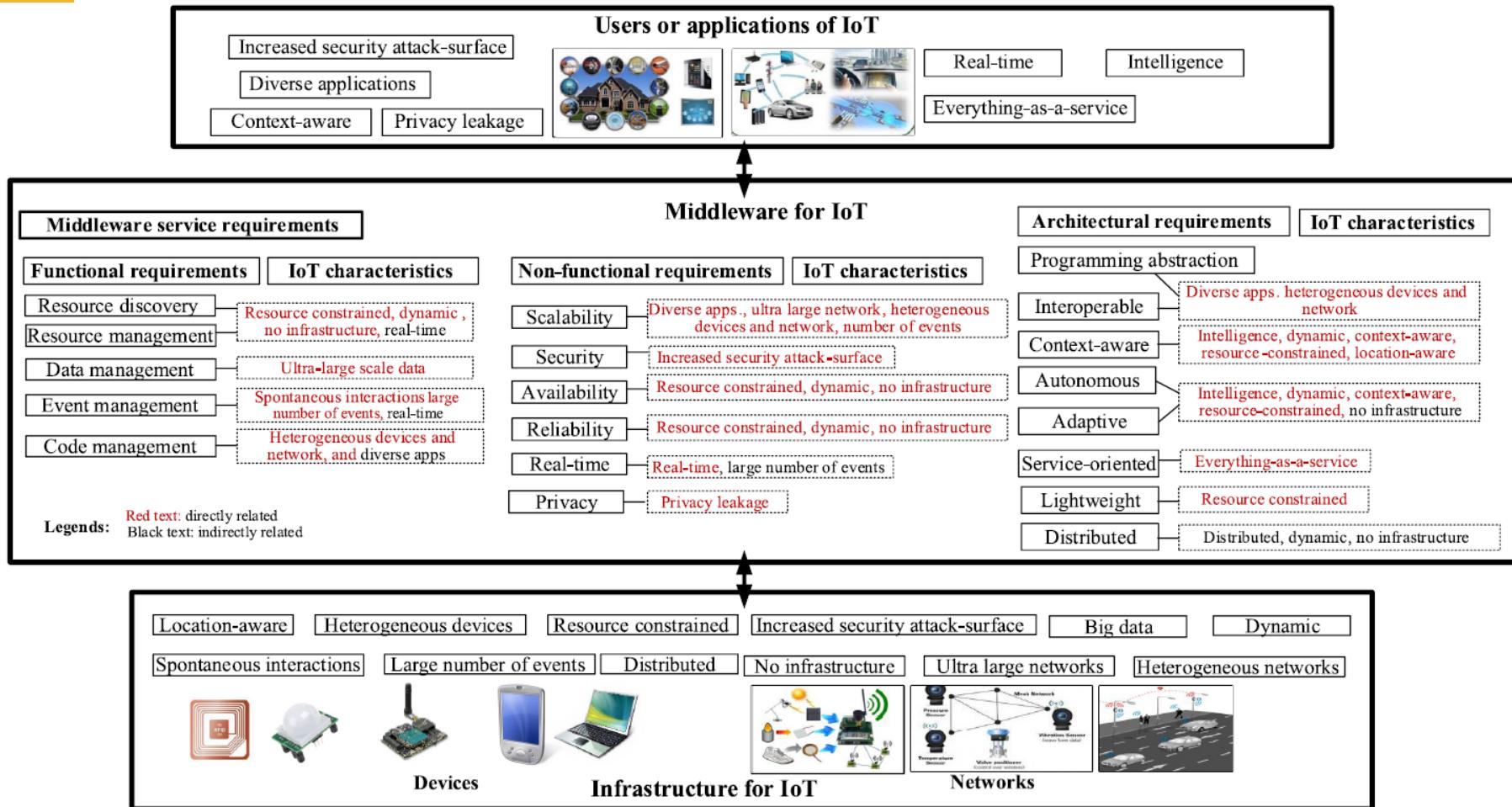
- WSN - se sastoji od prostorno raspoređenih autonomnih senzora za praćenje fizičkih ili okolišnih uvjeta, kao što su temperatura, pritisak, kretanje ili sl. te da proslijede svoje podatke kroz mrežu uglavnom kratkog dometa
- SCADA – autonomni sustav koji povezuje, prati i upravlja opremom putem mreže u objektu kao što je postrojenje ili zgrada

# Razlicitost IoT uređaja



Razzaque, Mohammad Abdur, et al. "Middleware for internet of things: a survey." *IEEE Internet of things journal* 3.1 (2015): 70-95.

# Zahtjevi za middleware



Razzaque, Mohammad Abdur, et al. "Middleware for internet of things: a survey." *IEEE Internet of things journal* 3.1 (2015): 70-95.

- Međuprogramska oprema, međusoftver, međuprogram?
- Pojam dolazi iz distribuiranog računalstva - skup servisa kao što su standardizirani API-ji, protokoli i infrastrukturne usluge za podršku brzom razvoju distribuiranih usluga i aplikacije temeljenim na klijent/poslužitelj i višeslojnim paradigmama
- Middleware se odnosi na integraciju i interoperabilnost aplikacija i usluga koje se izvode na heterogenim računalnim i komunikacijskim uređajima

- Termin *middleware* odnosi se na sloj koji je raspoređen iznad operativnih sustava i komunikacijskih elemenata i time skriva heterogenost od aplikacija kroz skup uobičajenih, dobro definiranih sučelja
- Na ovaj način, distribuirane klijentske i poslužiteljske komponente od kojih se sastoji aplikacija mogu se programirati na isti način kao da se izvršavaju samo na jednom računalu.
- Gartner definira middleware kao „Izvršni sistemski softver koji omogućava interakcije na razini programa u distribuiranom okruženju“

- Omogućuje međusobnu komunikaciju aplikacija koje se izvršavaju na različitim platformama
- Štite programere od ovisnosti o specifičnim mrežnim protokolima, operacijskim sustavima ili hardverskim platformama
- Skriva heterogenost i ovisnost o lokaciji
- Povećava prenosivost softvera
- Pomaže interoperabilnosti aplikacija
- Pomaže skalabilnosti
- Pruža uobičajene funkcionalnosti koje trebaju različitim aplikacijama
- Pomaže u integraciji sa starim (legacy) aplikacijama

- Message-Oriented Middleware (MOM/MQ/JMS/ESB)
- CEP (complex event processing) Middleware (Tibco, Sybase)
- Adaptive and Reflective Middleware (TAO/DynamicTAO/OpenORB)
- Transaction Middleware (TPM/Tuxedo)
- Peer-to-Peer Middleware (JXTA)
- Grid Middleware (PVM/MPI/Schedulers)
- Model-Driven Middleware (CoSMIC)
- Games Middleware (Autodesk)
- Mobile Computing Middleware (OSA/Parlay/JAIN/OMA)
- Radio-frequency Identification (RFID) (Smart Cards) Middleware (Edgeware)

- Three-tiered Application Server Middleware (Weblogic, Websphere)
- Real-time CORBA Middleware (Real-time CORBA)
- High-Availability (Fault Tolerance) Middleware (Fault-Tolerant CORBA)
- Security Middleware (Siteminder)
- Process-Oriented Middleware (WebMethods, SeeBeyond, Tibco, IBM, SAP, Oracle)
- Business-to- Business (B2B)-Oriented Middleware (SeeBeyond/Oracle, Tibco, webMethods)

Gartner prepoznaće tri glavne kategorije middleware-a:

- Integracijski middleware (integration middleware)
- Osnovni/temeljni middleware (basic middleware)
- Alati za razvoj i upravljanje

- Pokriva poslovne i aplikacijski-orientirane karakteristike poput:
- Upravljanje poslovnim procesima
- Radni tokovi poslovnih pravila
- Upravljanje poslovnim događajima
- Usmjeravanje podataka i adapteri

Temelj koji se odnosi i na samu IoT infrastrukturu:

- Middleware za upravljanje podacima
- Komunikacijski middleware
- Platformski middleware – pruža kontejner za aplikacijske komponente

Troslojni model DCM (device, connect, and manage) može se proširiti na više slojeva:

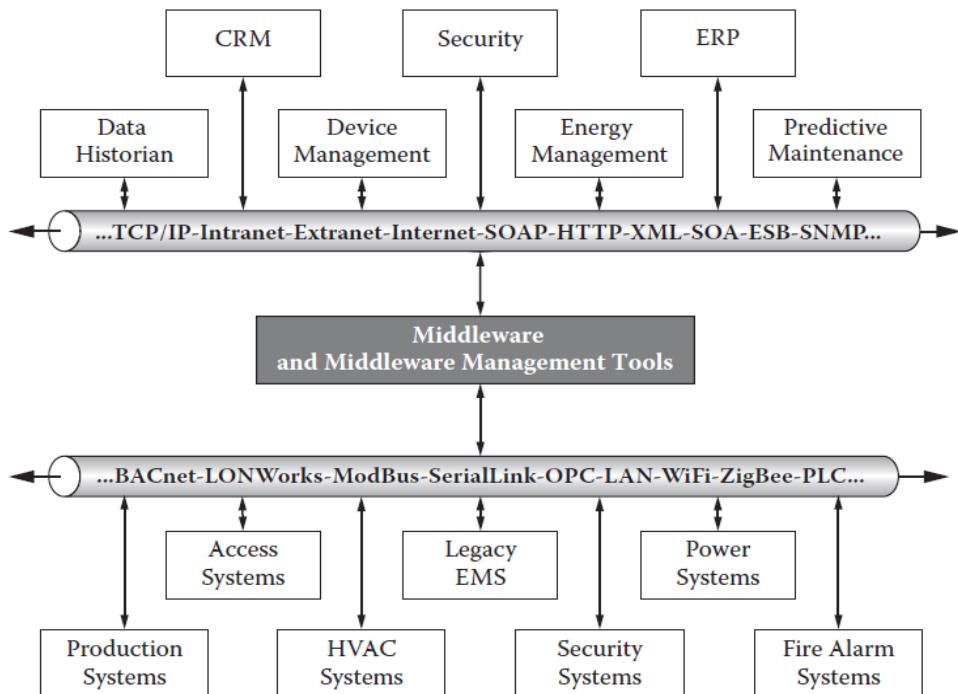
- Body (BAN)
- Personal (PAN)
- Near-me (NAN)
- Machine-to-machine, or M2M (MAN)
- Local (LAN)
- Home (HAN)
- Storage (SAN)
- Campus (CAN)
- Backbone
- Metropolitan (MAN)
- Wide (WAN)
- Internet

- Machine to machine
- 3GPP (Third Generation Partnership Project) je kolaboracija između grupa telekomunikacijskih udruženja (ETSI itd.).
- M2M platforma: M2M područna mreža (pruža žičanu ili bežičnu vezu između M2M uređaja i M2M pristupnika, kao što je osobna mreža); M2M pristup/jezgra mreža (međusobno povezivanje M2M uređaja od gatewaya do pristupa/jezgre kao što su GPRS/GSM, SGSN, WCDMA itd.); vanjske/Internet mreže (komunikacija između 3GPP pristupa/jezgre mreže i M2M međuslojne platforma za aplikacije, kao što su Internet, korporativni WAN-ovi i drugi)

- Satelitski generirani telemetrijski podaci
- Podaci o lokaciji kao što su očitanja RFID čipa, globalno pozicioniranje (GPS)
- Očitanja temperature i drugih senzora okoliša
- Očitanja senzora iz tvornica i cjevovoda
- Izlaz iz mnogih vrsta medicinskih uređaja, iz bolnica i domova

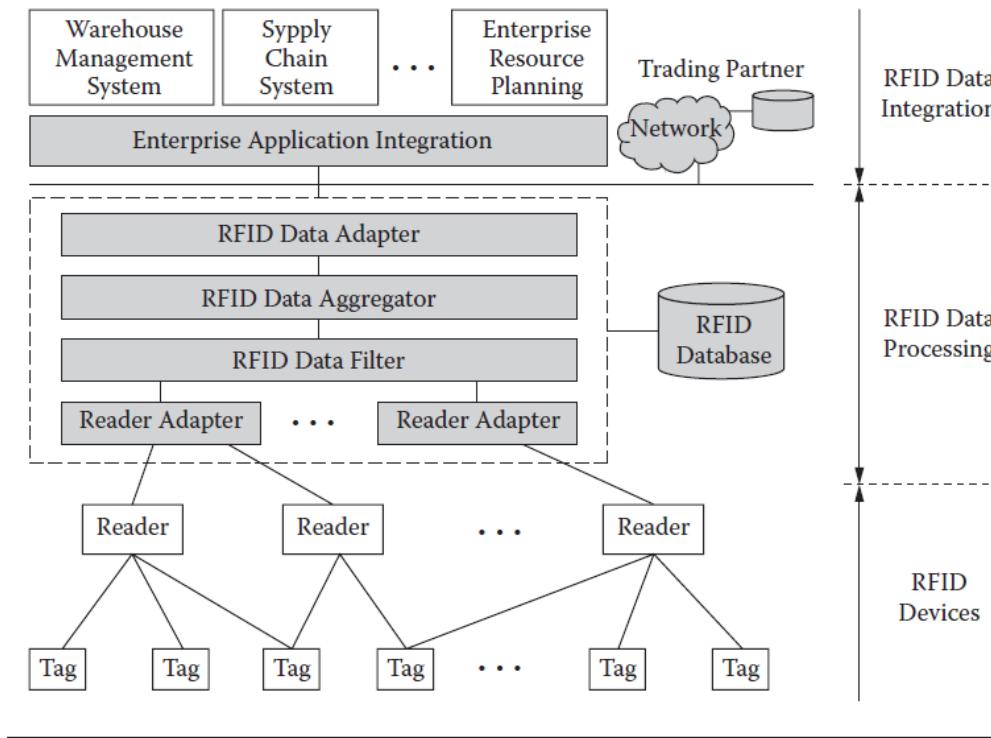
# SCADA middleware

- Udaljene terminalne jedinice (RTU), programabilne logički kontroleri (PLC-ovi), ili čak kontrola procesa sustavi (PCS-ovi) komuniciraju sa SCADA međuslužiteljem (middleware server) putem pristupnika (sličnih MAN-u, ali svi ožičeni) koji agregiraju podatke iz različitih žičnih sabirnica polja.



The Internet of Things in  
Cloud: Middleware  
Perspective

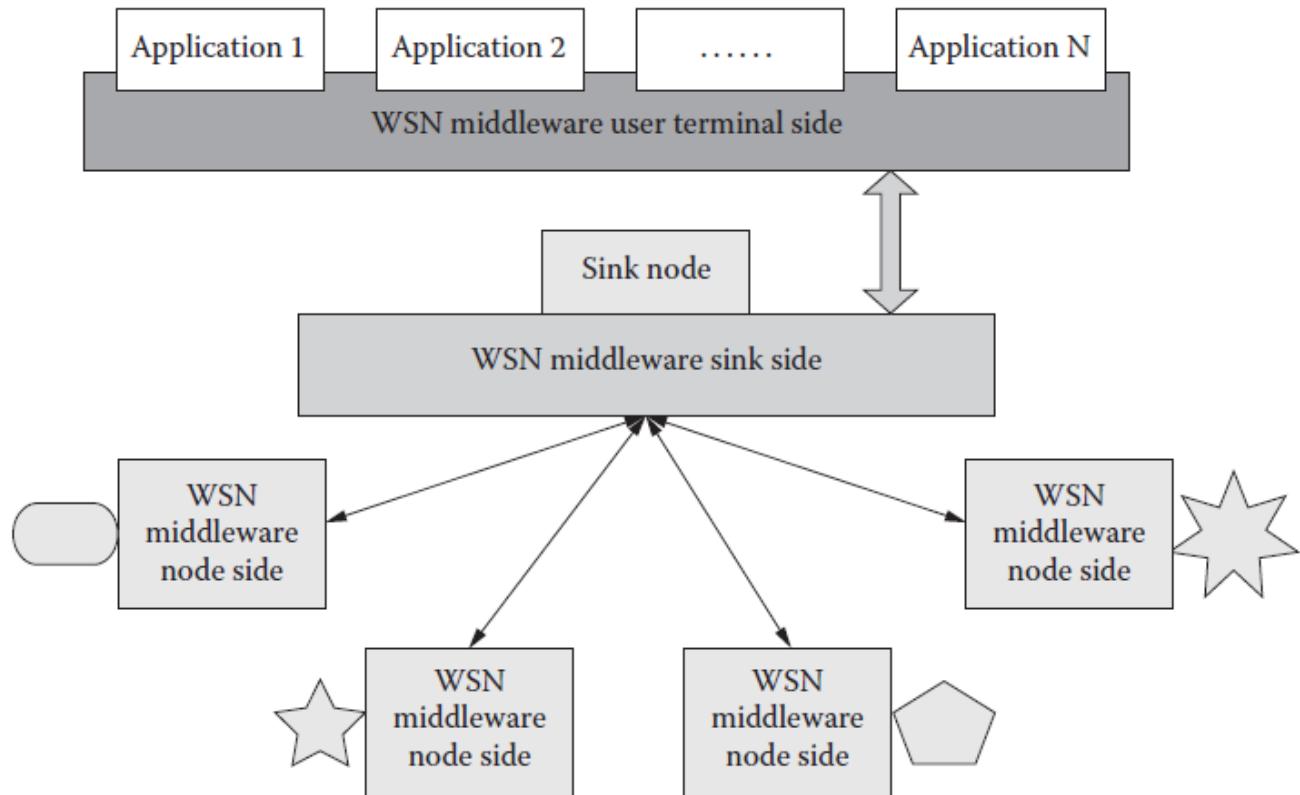
- RFID čitač -> tvrtkin LAN -> ako treba onda još i na Internet
- RFID middleware ie dobro definiran i standardiziran



## WSN middleware (1)

- Pružanje željene usluge za senzore sveprisutne računalne aplikacije koje koriste WSN i srodne ugrađene operativne sustave ili firmware senzorskih čvorova
- Mora biti svjestan konteksta
- Većinom je ugrađen na samom čvoru
- Cjelovito WSN middleware rješenje trebalo bi uključivati četiri glavne komponente: programske apstrakcije, sistemske usluge, runtime podršku i mehanizme kvalitete usluge (QoS).

# WSN middleware (2)

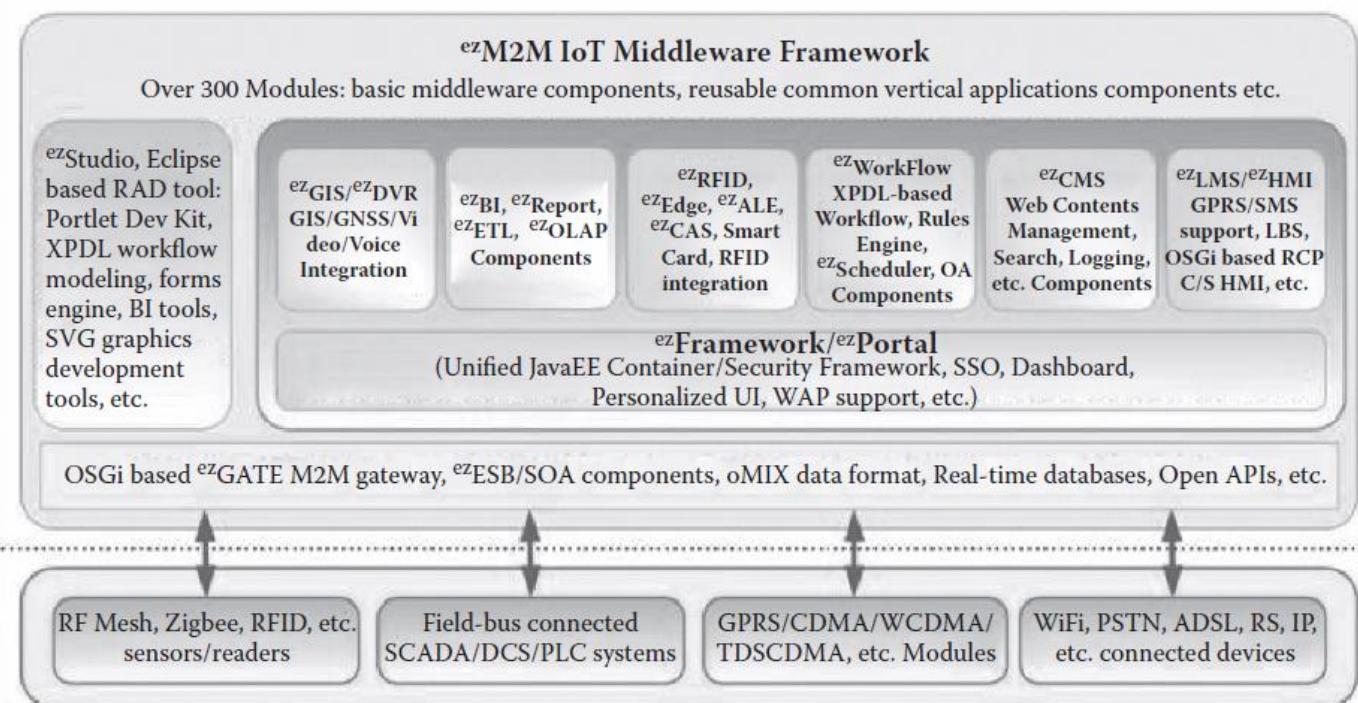


- Trenutna tržišta Interneta stvari (IoT) i Web of Stvari (WoT) su jako fragmentirana.
- Razni vertikalna WoT/IoT rješenja su dizajnirana neovisno i zasebno za različite primjene, što neizbjježno utječe ili čak otežava implementacija WoT većeg opsega.
- Jedinstvna, horizontalna platforma temeljena na standardima je ključ za konsolidaciju fragmentacije.
- Platformski middleware se često naziva i aplikacijskim okvirom za IoT

- Glavni cilj platformskog middleware-a je približiti IoT aplikacije web, tj. ostvariti web stvari
- Primjer – mangOH - <https://mangoh.io/> - open source IoT end-to-end platform
- M2M platformski middleware – uobičajeno pokriva razine od M2M gatewaya do M2M aplikacijskog poslužitelja – npr. Actility - <https://www.actility.com/>
- Postoje mnogi istraživački projekti koji se bave IoT middleware-om

- OSGi (Open Services Gateway initiative framework)
- je modularni sustav i servisna platforma za Java programski jezik koji provodi cjelovit i dinamičan komponentni model.
- Korištenje OSGi-ja, aplikacija ili komponente (dolaze u obliku paketa za implementaciju) mogu se udaljeno instalirati, pokrenuti, zaustaviti, ažurirati, i deinstalirati bez potrebe za ponovnim pokretanjem.
- Upravljanje Java paketima/klasama specificirano je vrlo detaljno.
- OSGi specifikacije su se pomaknule dalje od originalnog fokusa na service gatewayu i sada se koriste u aplikacijama u rasponu od mobilnih telefona do Eclipse IDE-a(koji dominira tržištem IDE).

# Primjer: ezM2M middleware koji koristi OSGi



# Višeslojni IoT middleware

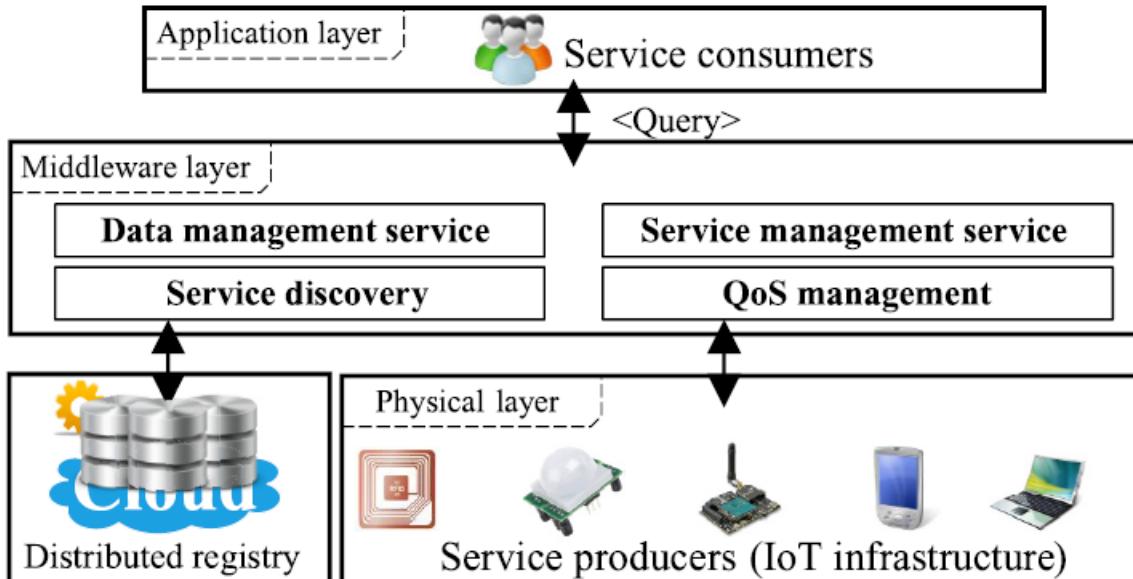
Multi-tiered IoT Middleware		IoT Graphics/HMI RAD Tools, Reporting, Trending, Data Mining, Decision Support, etc.			
	Service Oriented Middleware Layer	Business Oriented Component (BPM, Workflow/Rule Engine, Content Management, multi-tenancy, SOA/EAI, etc.)			
	Basic Middleware Component Layer	Application Server (Websphere, WebLogic, Jboss, .NET Framework/IIS, etc.) OSGi Framework, etc.)			
	IoT Connectivity Middleware Layer	M2M Gateway, JCA/Adaptors (OPC, GPRS, Field-bus, etc.) MQ/ESB/JMS, open API, etc.			
DBMS Layer	Database (Oracle, IBM, SQL Server, mySQL, etc.) Real-time Databases, etc.				
Hardware, OS (Linux, Unixes, Windows, etc.) and Networks					

The Internet of Things in Cloud: Middleware Perspective

- Middleware rješenja grupirana prema pristupima dizajna:
- Middleware baziran na događajima
- Servisno-orientirani middleware
- Middleware baziran na virtualnim strojevima
- Agentno bazirani middleware
- Tuple-spaces middleware
- Middleware orijentiran na baze podataka
- Middleware specifičan za primjenu (application-specific)

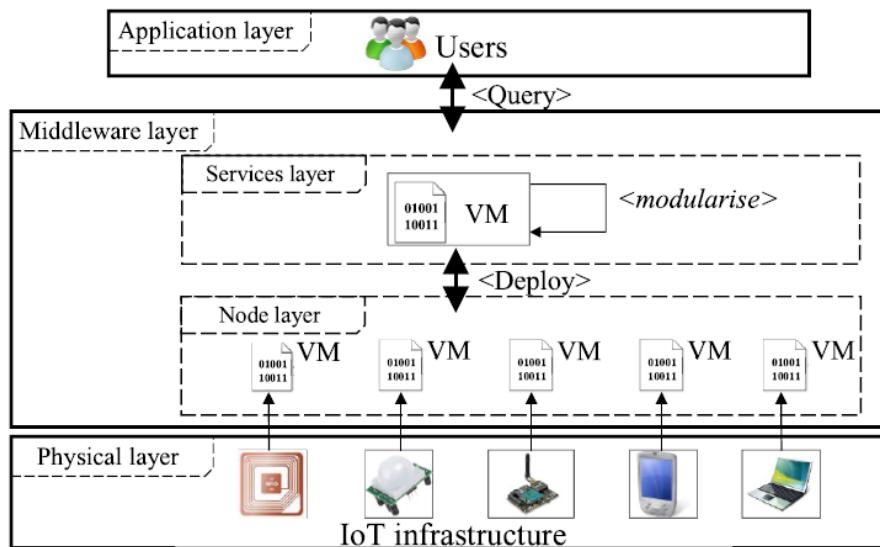
- Komponente, aplikacije i svi ostali sudionici komuniciraju kroz događaje
- Svaki događaj ima tip, kao i skup tipiziranih parametara čije specifične vrijednosti opisuju specifičnu promjenu stanja proizvođača.
- Događaji se prenose iz komponenti aplikacije koje šalju (proizvođači), primateljskim komponentama aplikacije (potrošačima).
- Sustav događaja (usluga događaja) može se sastojati od potencijalno velikog broja aplikacijskih komponenti (entiteta) koje proizvode i konzumiraju događaje
- Obično koriste publish/subscribe predložak – obavijesti o događajima se asinkrono šalju preplatnicima

- Aplikacije/softver se gradi u obliku servisa
- Karakteristike servisno-orientiranog pristupa korisne su i za IoT – neutralnost od tehnologije, labavo spajanje, ponovna iskoristivost servisa, spajanje i otkrivanje servisa
- Ipak zbog karakteristika IoT-a, otkrivanje i spajanje servisa je izazovno



Razzaque, Mohammad Abdur,  
et al. "Middleware for internet  
of things: a survey." *IEEE  
Internet of things journal* 3.1  
(2015): 70-95.

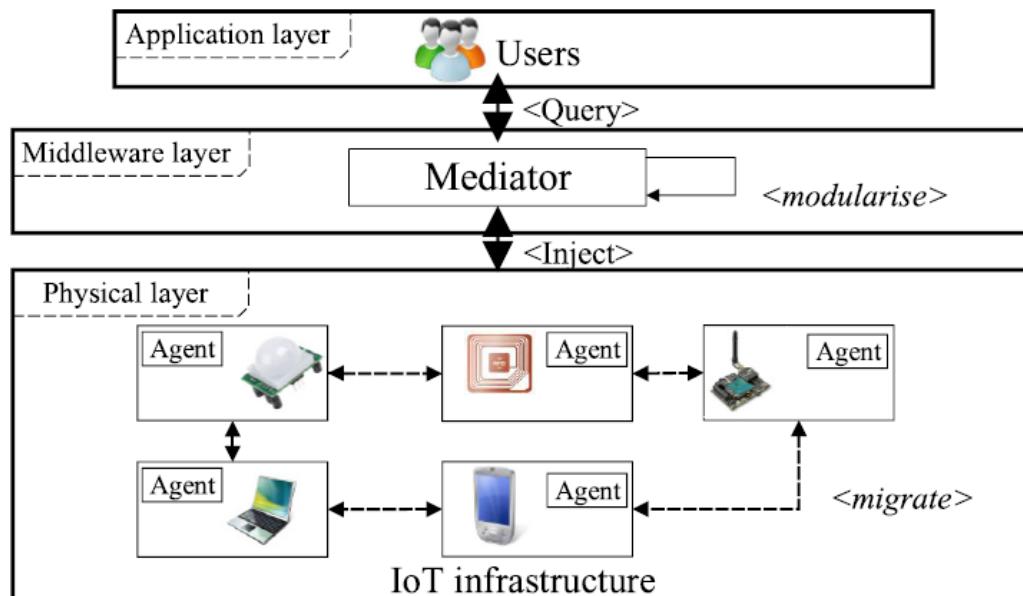
- Pruža programsku podršku za sigurno izvršno okruženje za korisničke aplikacije virtualizacijom infrastrukture.
- Aplikacije su podijeljene u male zasebne module koji se distribuiraju kroz cijelu mrežu.
- Svaki čvor u mreži sadrži VM koji tumači module



Razzaque, Mohammad Abdur,  
et al. "Middleware for internet  
of things: a survey." *IEEE  
Internet of things journal* 3.1  
(2015): 70-95.

# Agentno bazirani middleware

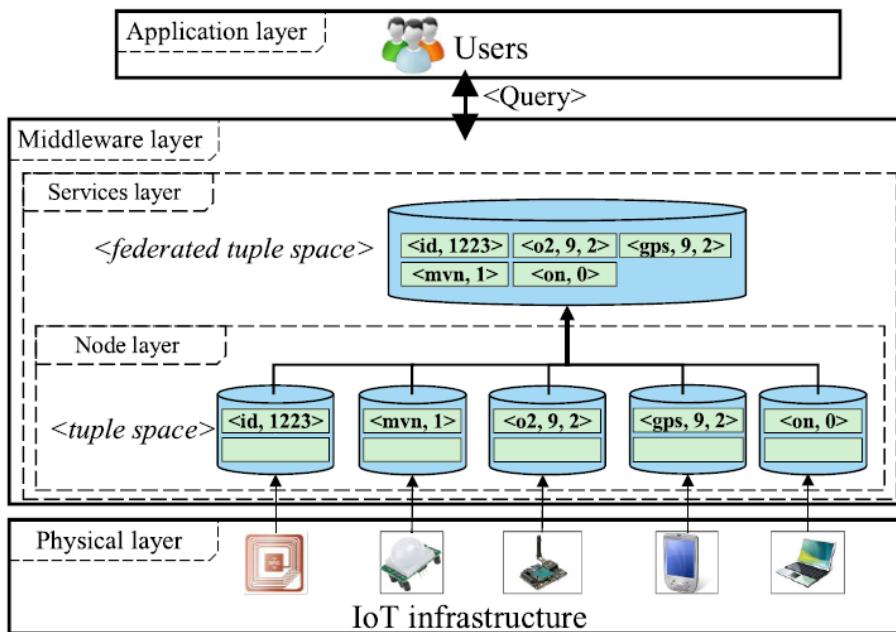
- Aplikacije su podijeljene u modularne programe koji se distribuiraju kroz mrežu korištenjem mobilnih agenata.
- Dok migriraju s jednog čvora na drugi, agenti čuvaju njihovo stanje izvršavanja.
- Ovo olakšava dizajn decentraliziranih sustava sposobnih tolerirati djelomične kvarove.



Razzaque, Mohammad Abdur,  
et al. "Middleware for internet  
of things: a survey." *IEEE  
Internet of things journal* 3.1  
(2015): 70-95.

# Tuple-spaces middleware

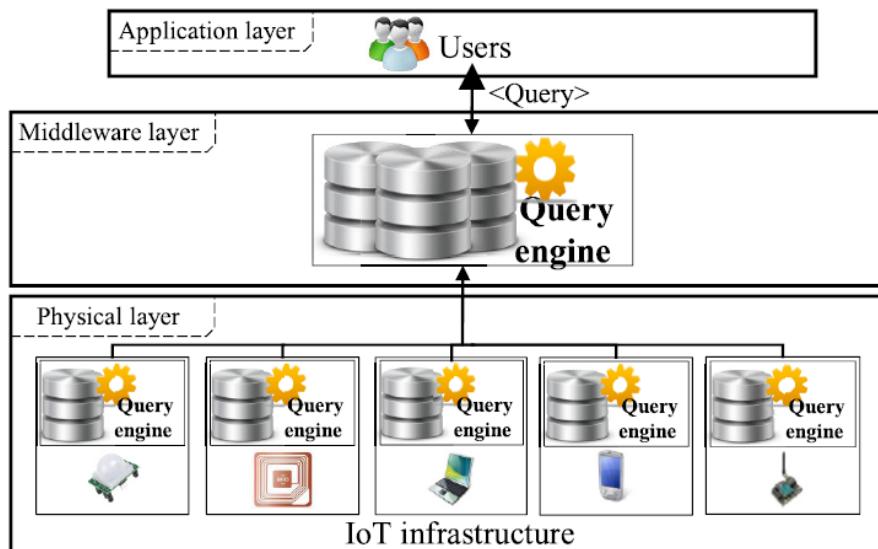
- Svaki član infrastrukture sadrži lokalnu tuple-space strukturu.
- Tuple space – repozitorij podataka kojemu se može konkurentno pristupiti
- Svi tuple spaces formiraju federalni tuple-space na gatewayu



Razzaque, Mohammad Abdur,  
et al. "Middleware for internet  
of things: a survey." *IEEE  
Internet of things journal* 3.1  
(2015): 70-95.

# Middleware orientiran na baze podataka

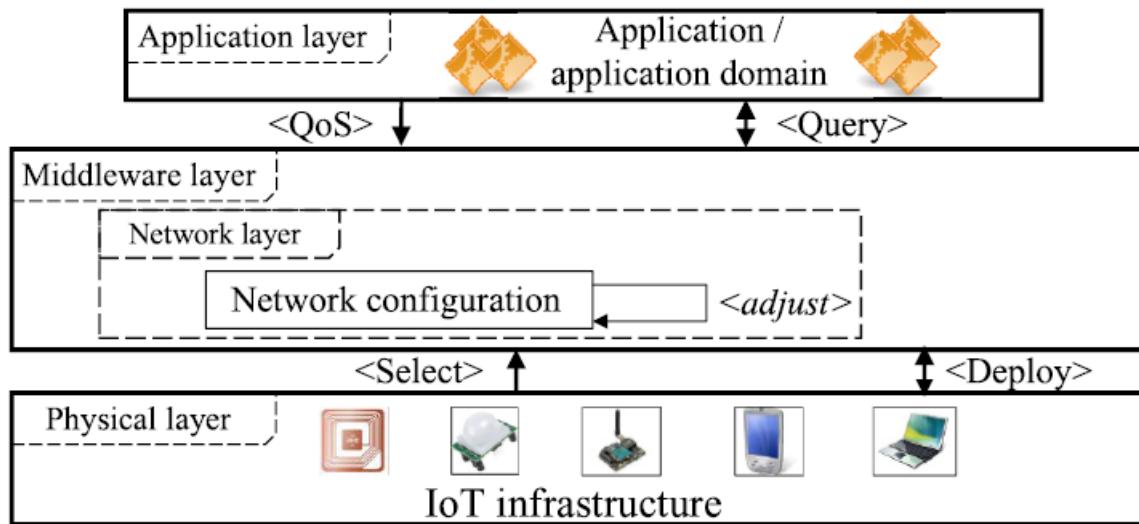
- Mreža senzora se promatra kao virtualna relacijska baza podataka.
- Aplikacija može bazi podataka postavljati upite slične SQL-u – mogućnost formulacije kompleksnih upita
- Pristup distribuiranih baza podataka za interoperabilnost sustava



Razzaque, Mohammad Abdur,  
et al. "Middleware for internet  
of things: a survey." *IEEE  
Internet of things journal* 3.1  
(2015): 70-95.

# Middleware specifičan za primjenu

- Fokusira se na podršku upravljanja resursima (tj. QoS) za određenu primjenu ili domenu primjene implementacijom arhitektura koja fino podešava mrežu ili infrastrukturu na temelju primjene ili prema domenskim zahtjevima specifične primjene.



Razzaque, Mohammad Abdur,  
et al. "Middleware for internet  
of things: a survey." *IEEE  
Internet of things journal* 3.1  
(2015): 70-95.

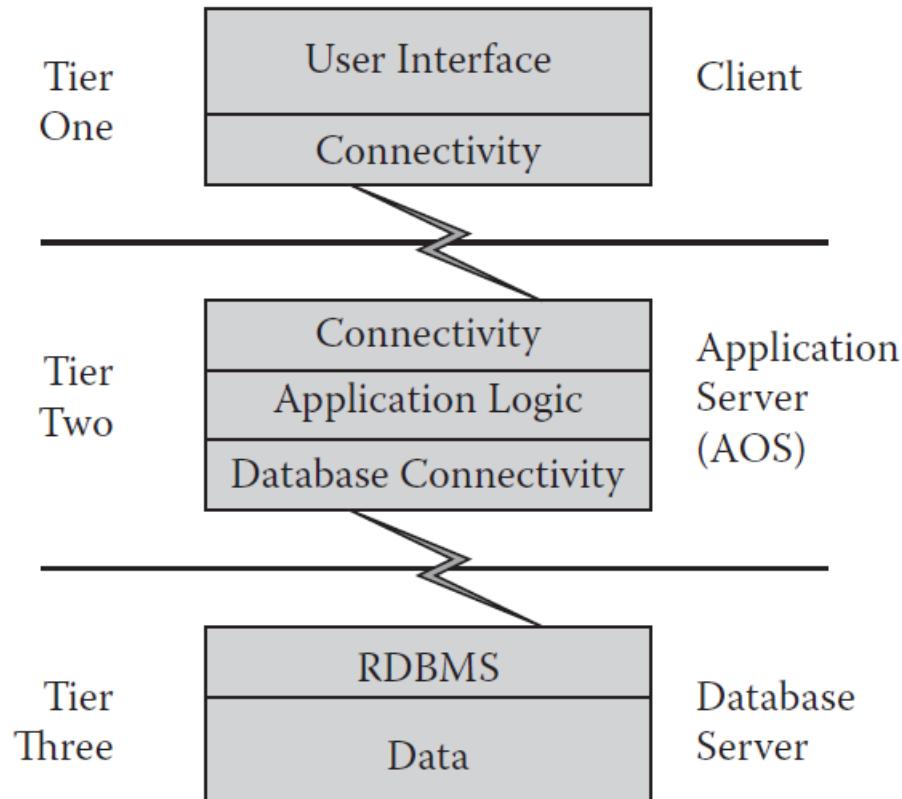


# Standardizacija protokola za IoT

Darko Andročec

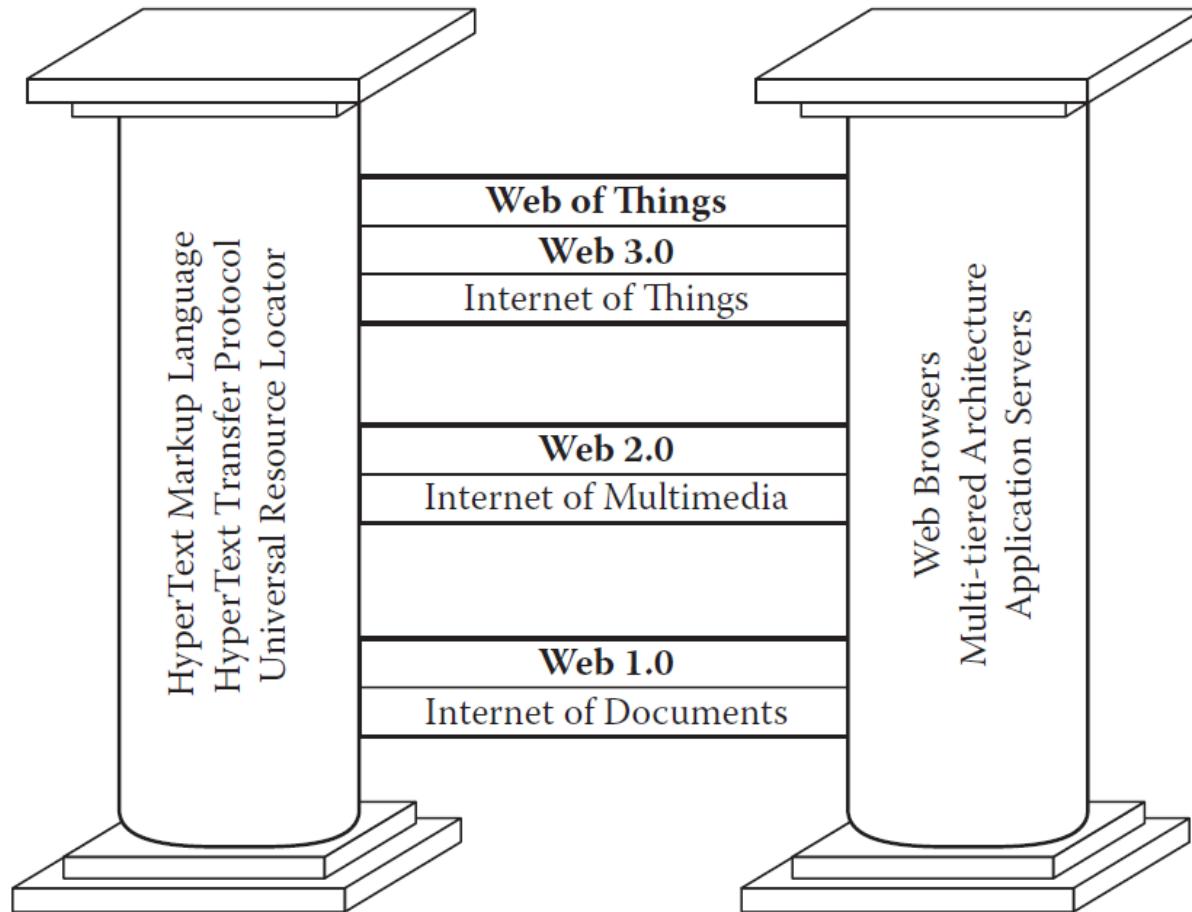
- WWW je najpopularnija metoda korištenja Interneta
- Web stvari bi trebao postati glavna aplikacijska platforma ili osnova IoT-a
- Globalna mreža senzora i aktuatora koja omogućuje nove primjene i funkcionalnosti
- Prihvaćeni standardi poput HTTP, Restfull API i sl. se koriste za pristupanje funkcionalnostima pametnih stvari

# Troslojna arhitektura



The Internet of Things in Cloud: Middleware Perspective

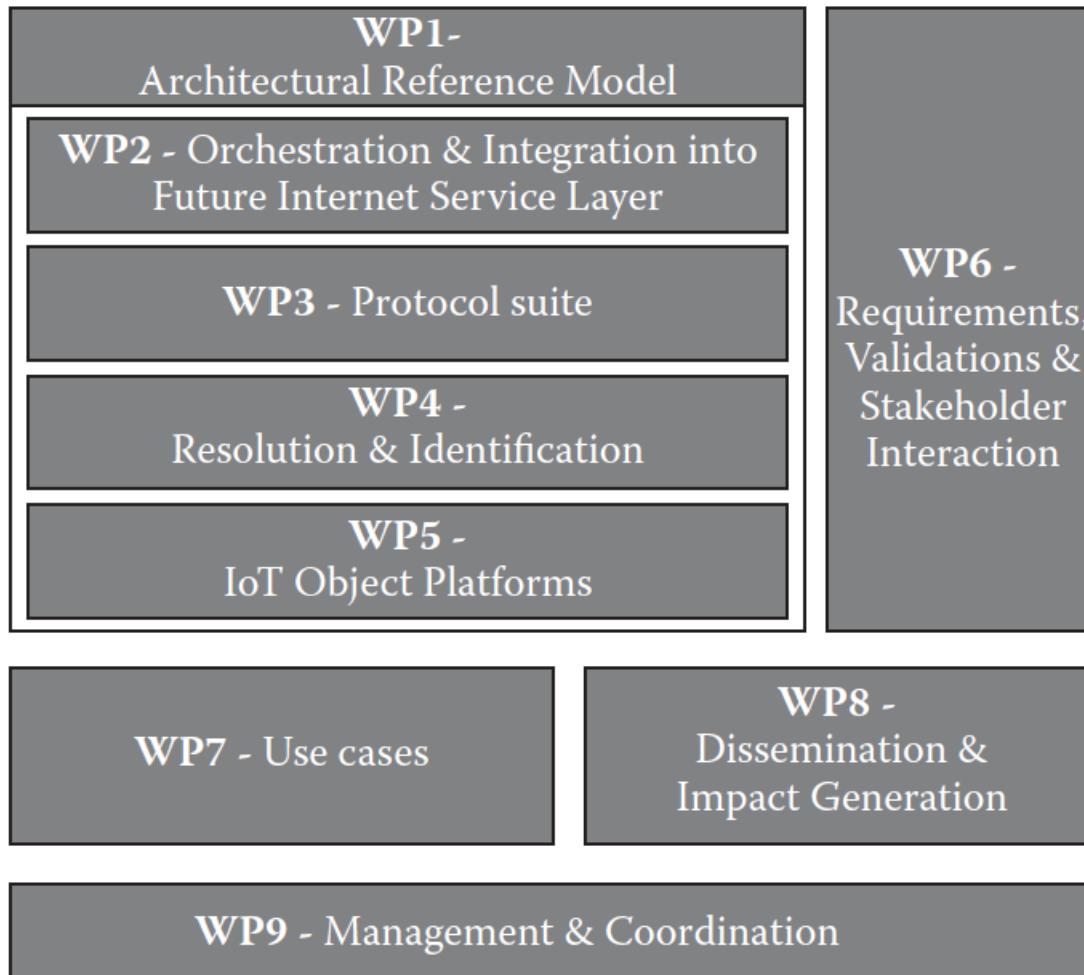
# Dva stupá weba



The Internet of Things in Cloud: Middleware Perspective

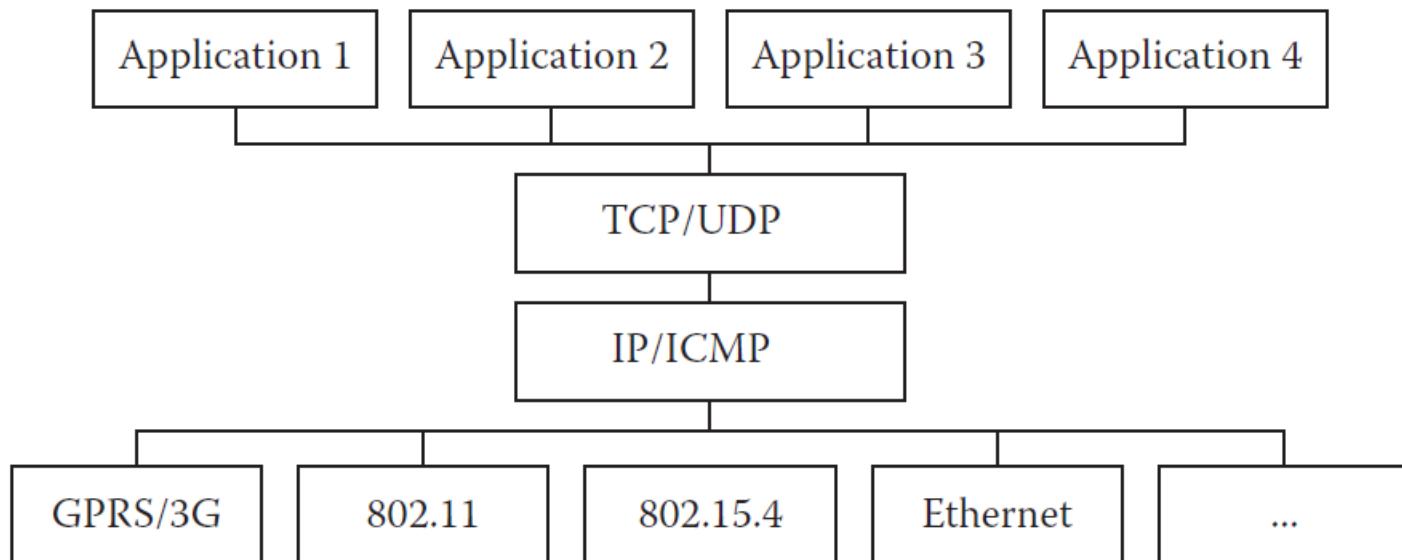
- Fragmentirane arhitekture, bez koherentnih unificirajućih koncepata, rješenja postoje samo za silose primjene.
- Još ne postoji holistički pristup implementaciji IoT-a
- Postoje mnoga djelomična pojedinačna rješenja (RFID, senzorske mreže)
- Malo je ponovne uporabe tehnologije i razmjene znanja između različitih sektora

- IoT-A (Internet of Things architecture) konzorcij se sastoji od 17 europskih organizacija iz 9 zemalja
- Glavni cilj: kreiranje arhitekturne osnove interoperabilnog interneta stvari kao ključne dimenzije Interneta budućnosti
- Spajanje heterogenih IoT tehnologija u interoperabilno IoT okruženje



# IPSO (Internet Protocol for Smart Objects)

- Kako koristiti IP za IoT pametne objekte temeljene na sve-IP holističkom pristupu.



The Internet of Things in Cloud: Middleware Perspective

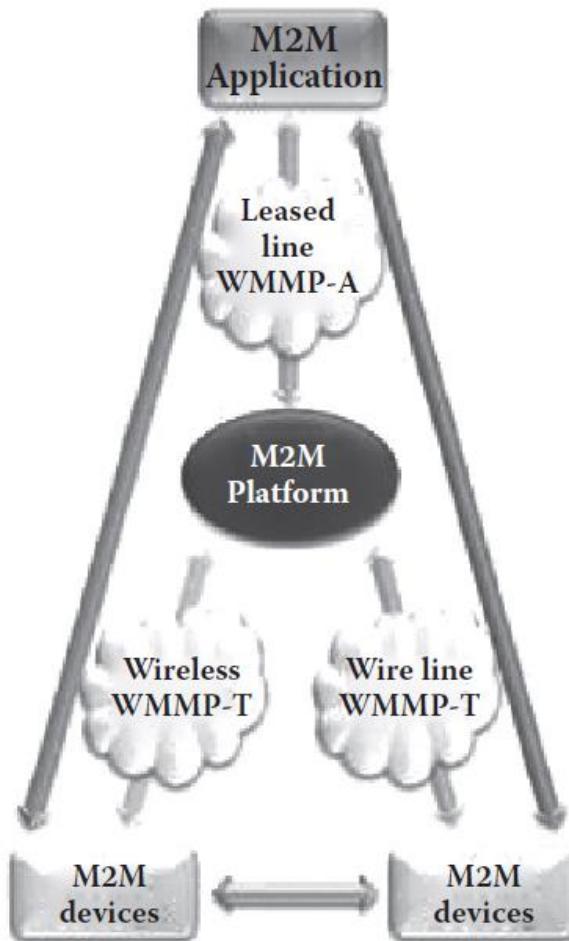
- Inteligentna (RFID) oznaka
- Senzor – uređaj koji mjeri fizičku kvantitetu i pretvara je u digitalni ili analogni signal
- Aktuator – uređaj koji kontrolira skup opreme
- Ugrađeni uređaj – uređaj koji izvodi specifičnu funkciju, recimo robotska ruka u proizvodnji
- Bilo koja kombinacija gornjih funkcionalnosti koja tvori složeniji entitet

- IPv4, IPv6, 6LoWPAN
- Mobile IP protokol

- International Telecommunication Union's (ITU) i ETSI's (M2M Technical Committee) Global Standards Collaboration (GSC) uspostavili su M2M Standardization Task Force (MSTF)
- Krajnji rezultat ovih napora je definiranje pojmovnog okvira za M2M aplikacije za vertikalna industriju i agnostičku za komunikacijske tehnologije i napraviti sloj usluge koji će programerima aplikacija omogućiti stvaranje aplikacija koje djeluju transparentno u različitim vertikalama domene i komunikacijske tehnologije bez da programeri moraju napisati vlastiti složeni prilagođeni sloj usluge.

- Standardi za protokole za prijenos podataka: M2MXML, JavaScript Object Notation (JSON), BiTXML, WMMP MDMP, open Building Information Exchange (oBIX), EEML, open M2M Information exchange (oMIX)
- Proširenje OMA DM za podršku M2M uređaja
- Upravljanje M2M uređajima, standardizacija M2M getewaya
- Sigurnost M2M-a i detekcija prijevara
- Daljinska dijagnostika i nadzor, daljinsko pružanje i otkrivanje
- Daljinsko upravljanje uređajima iza pristupnika ili vatrozida
- Otvoreni API temeljen na REST-u za M2M aplikacije

# Primjer: kineski mobilni WMMP standardi



## China Mobile WMMP Standards

### Version 1.0

- Enable the connection of devices to CMCC M2M Platform
- Allow users to manage and monitor M2M terminals in real time
- Based on Ericsson WMMP 0.98 platform

### Version 2.0

- Further refine platform/device protocol
- Define business and management flows
- Revise parts of v1.0 contents, enhance M2M device management functions
- Based on Chongqing Radium M2M 2.0 platform

### Version 3.0

- Include preamble definition, transfer process, device management, SIM card management, AT instruction format, etc.
- Realize device-to-device communications routed over M2M platform
- Include related security measures
- Add application API to platform
- Add device real time software upgrade function
- Further revise v2.0 contents
- Trial stage, platform providers: ZTE, Radium, device: SIMCOM, Huawei

### Future revision

- Support wireless sensor network technology
- Support non-standard and non-dedicated network and their protocols
- Support distributed M2M network access
- Realize automated terminal capacity distribution
- Realize direct device-to-device communications

- IEEE se fokusira na fizički i MAC sloj; IETF radi na slojevima 3 i višim
- IEEE 1451 je skup standarda sučelja pametnih pretvarača koje je razvio IEEE - skup otvorenih, zajedničkih, o mreži neovisnih komunikacijskih sučelja za povezivanje pretvarača (senzora ili aktuatora) s mikroprocesorima, instrumentacijskim sustavima i mrežama upravljanja/polja – bazirani su na XML-u

- Pristup označavanja podataka senzora s prostornim, vremenskim i tematskim semantičkim metapodacima koji se temelji na OGC SWE (Sensor Web Enablement)
- SWE Common—uobičajeni podatkovni modeli i shema
- SensorML—modeli i sheme senzorskih sustava i procesa koji okružuju mjerjenja
- Observations & Measurements (O&M) - modeli i shema za vrijednosti promatranja pakiranja
- Transducer Markup Language (TML)—modeli i shema za multipleksirane podatke iz senzorskih sustava

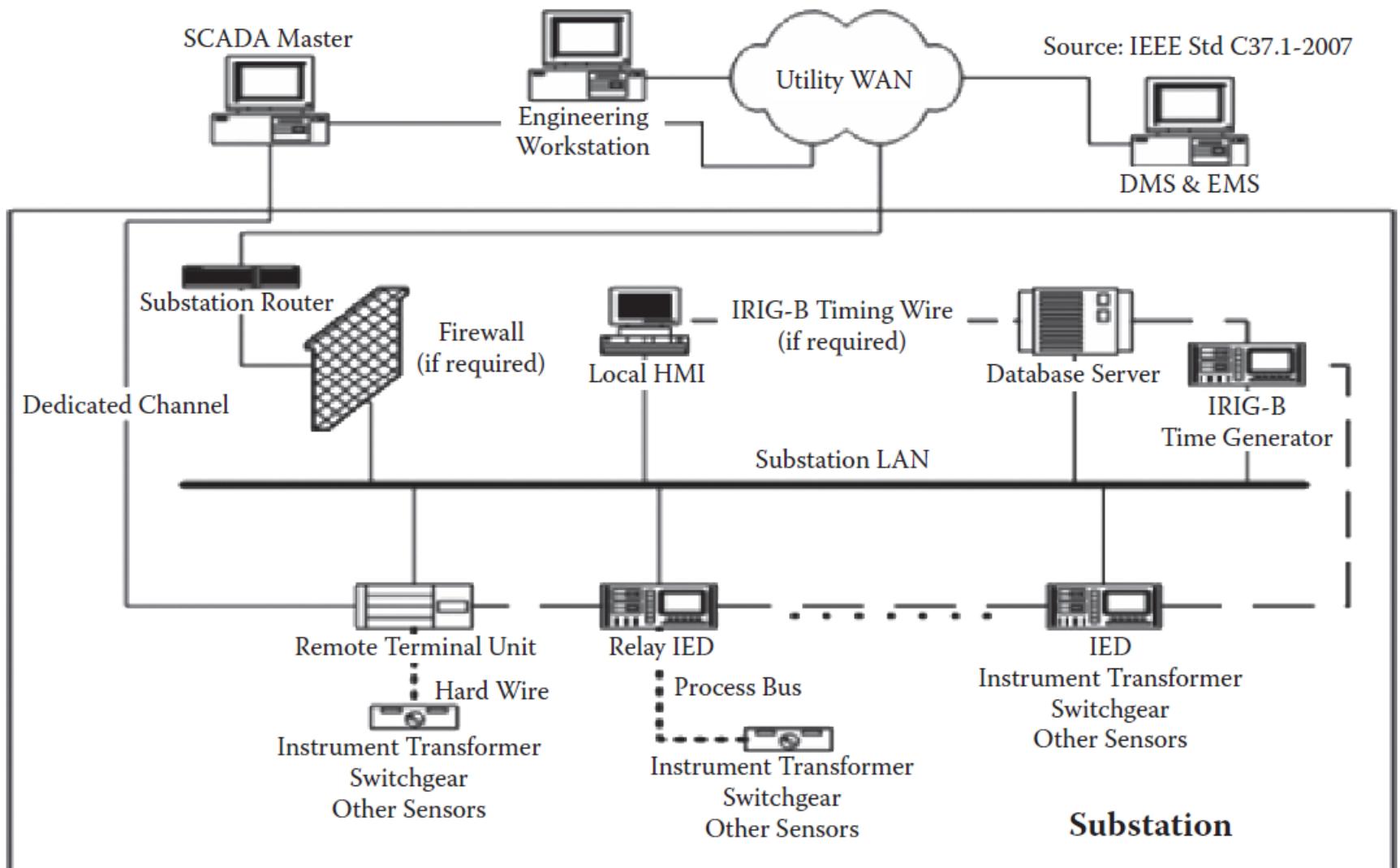
- EU projekt
- Stvara otvorenu, poslovno vođenu arhitekturu koja temeljno rješava probleme skalabilnosti velikog broja globalno distribuiranih bežičnih senzorskih i aktuatorских mreža (WSAN) uređaja
- Pruža potrebne usluge upravljanja mrežom i informacijama kako bi se omogućilo pouzdano i točno pronalaženje kontekstnih informacija i interakcija s fizičkim okruženjem.
- Dodavanjem mehanizama za računovodstvo, sigurnost, privatnost i povjerenje, omogućuje otvoren i siguran tržišni prostor za svijest o kontekstu i interakciju u stvarnom svijetu.

- Visoko skalabilni arhitektonski okvir s odgovarajućim rješenjima protokola koji omogućuju jednostavnu plug-and-play integraciju velikog broja globalno distribuiranih WSAN-ova u globalni sustav, pružajući podršku za upravljanje mrežom i informacijama, sigurnost, privatnost i povjerenje te računovodstvo.
- Otvoreno servisno sučelje i odgovarajuća semantička specifikacija za objedinjavanje pristupa kontekstualnim informacijama i uslugama pokretanja koje nudi sustav za usluge i aplikacije.
- Paneuropska platforma za testiranje

- Napravili su referentnu arhitekturu ISO/IEC 29182 za aplikacije senzorskih mreža i usluge usmjerenе na telekomunikacije i razmjenu informacija između sustava.
- ISO/IEC 29182 Part 1: General overview and requirements
- ISO/IEC 29182 Part 2: Vocabulary/terminology
- ISO/IEC 29182 Part 3: Reference architecture views
- ISO/IEC 29182 Part 4: Entity models
- ISO/IEC 29182 Part 5: Interface definitions
- ISO/IEC 29182 Part 6: Application profiles
- ISO/IEC 29182 Part 7: Interoperability guidelines

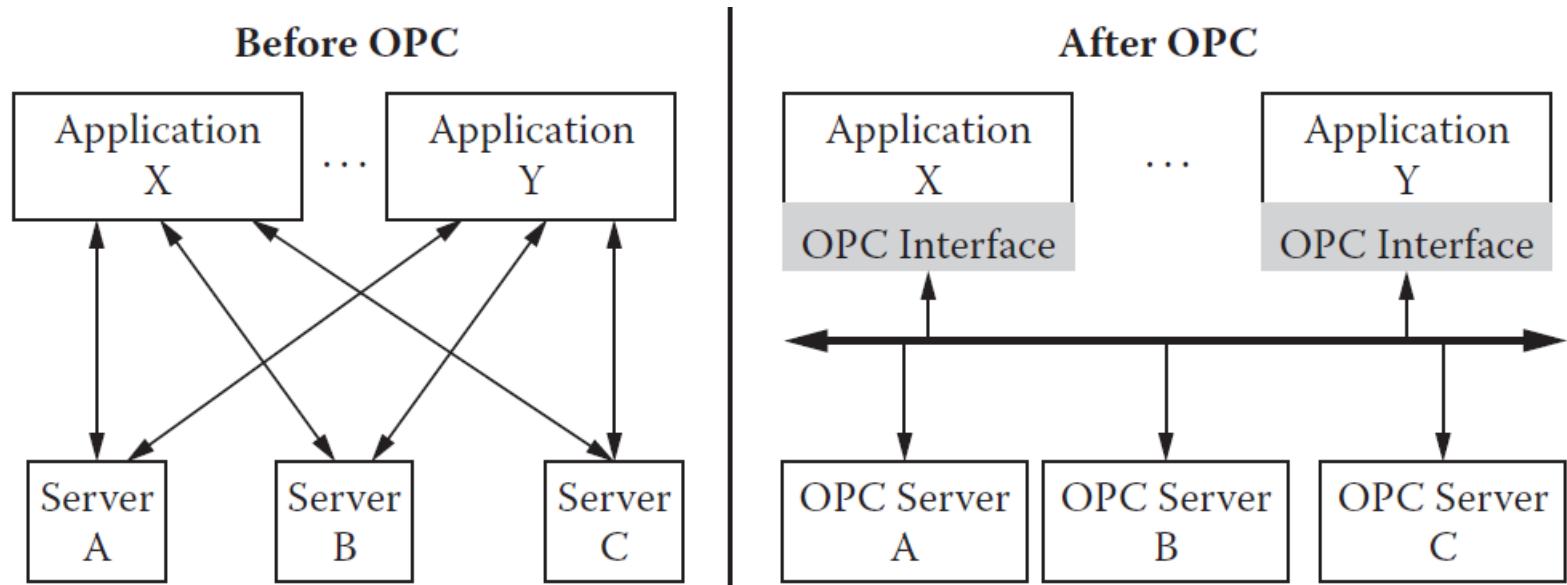
- SCADA – područje industrijske automatizacije
- Industrijska automatizacija ima različita vertikalna tržišta, pa postoje i mnoge vrste SCADA-a
- IEEE je stvorio standardnu specifikaciju, nazvanu Std C37.1™, za SCADA i sustave automatizacije 2007. godine

# IEEE Std. C37.1 SCADA architecture



- ISA100 = Wireless Systems for Industrial Automation
- Iako je bežična tehnologija napravila veliki korak naprijed s procvatom bežičnih osobnih komunikacija, aplikacije za industrijske sustave terenskih uređaja moraju se suočiti s izrazito različitim izazovima.

- OPC = Object Linking and Embedding (OLE) for Process Control
- Komunikacija podataka u realnom vremenu između kontrolnih uređaja različitih proizvođača



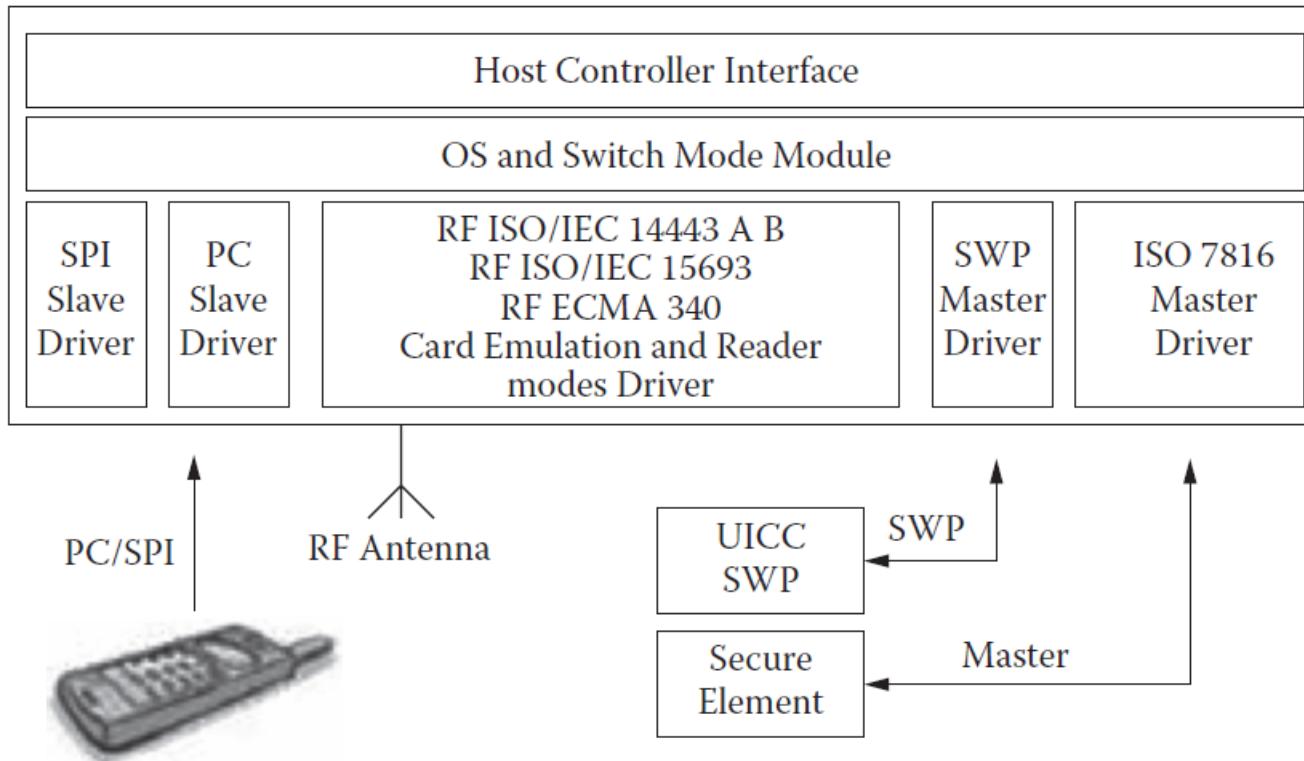
- Kada je proizvođač hardvera razvio svoj OPC poslužitelj za novi hardverski uređaj, njihov posao je obavljen kako bi bilo tko mogao pristupiti svom uređaju
- Kada je proizvođač SCADA razvio svoj OPC klijent, njihov posao je obavljen kako bi se omogućio pristup bilo kojem hardveru, postojećem ili tek stvorenom, s OPC-kompatibilnim poslužiteljem.
- OPC je postigao veliki uspjeh u mnogim područjima primjene, od kojih je većina usko povezana s IoT aplikacijama ili su dio njih.

## Nedostatak OPC-a

- Baziran je na Microsoftovoj COM i DCOM tehnologiji, pa je ograničen na operacijski sustav Windows
- Novi pristupi, kao što su XML-DA i United Architecture (UA), razvijeni su kako bi OPC tehnologija bila dostupna na drugim platformama ili drugim sustavima.

- Relativno su dobro definirani – većinom od EPCglobal
- PML, Object Naming Service (ONS), Edgeware, EPC Information Service (EPCIS), Application Level Event (ALE)

- Standard za beskontaktnu komunikaciju pametnih kartica



- Nije sve vezano uz standardizaciju pozitivno
- Standardizacija je ključna za razvoj tržišta, ali može ugroziti inovacije i spriječiti promjene kada tržište prihvati standarde
- Različiti konzorciji, forumi i savezi provode standardizaciju u svom ograničenom djelokrugu pokrivajući uglavnom područje koje im je poznato.
- Čak i unutar istog segmenta, postoji više od jednog konzorcija ili foruma koji rade standardizaciju bez dovoljno međusobne komunikacije, a neki se čak međusobno natječu.

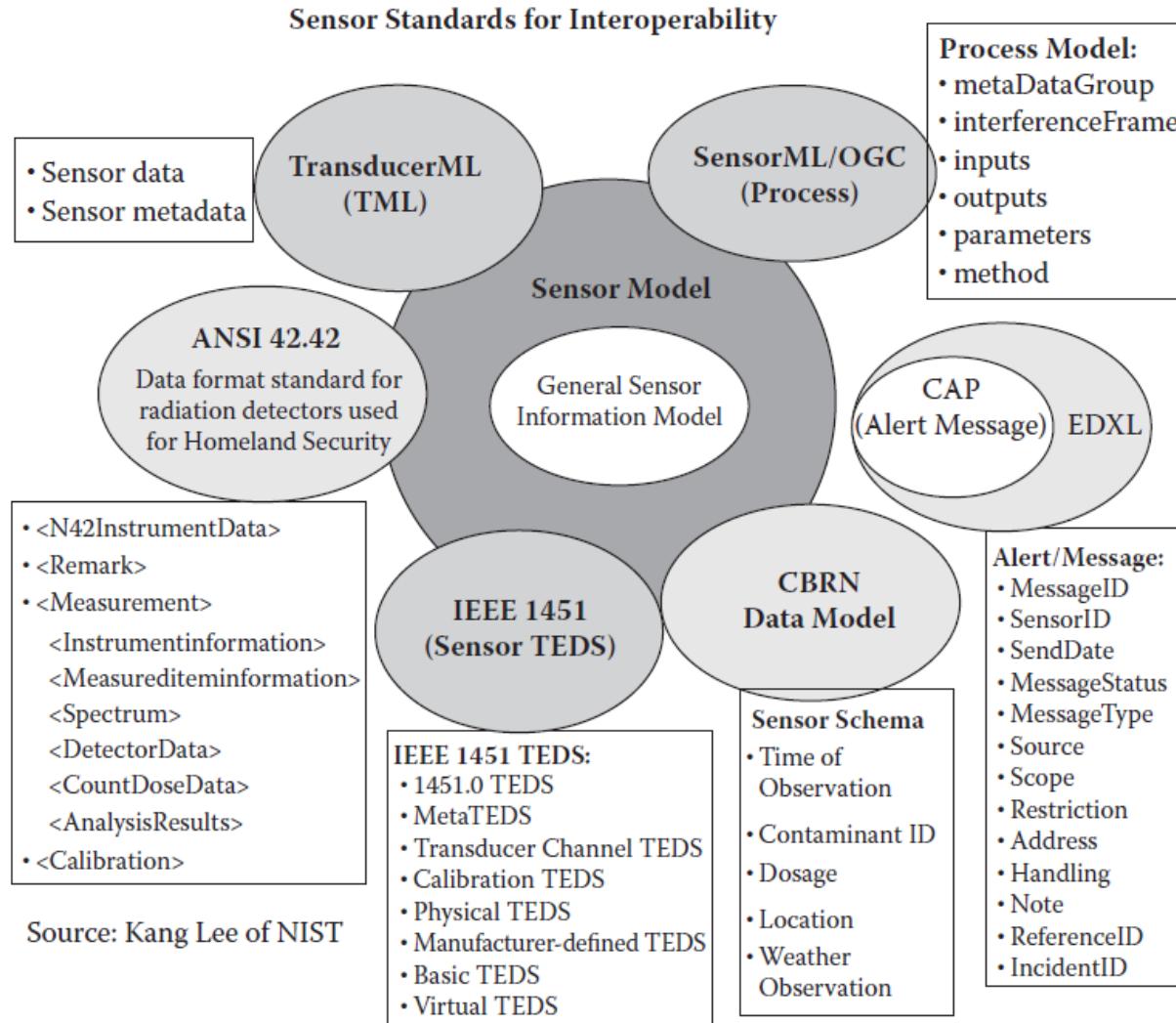
- Neki ljudi vjeruju da je IoT koncept dobro utemeljen; međutim, neke sive zone ostaju u definiciji
- Iako neke organizacije standarda IoT-a imaju suradnju i interakciju, ona je ograničena i nedovoljno otvorena.
- ICT standardizacija je visoko decentralizirana aktivnost. Kako se mogu koordinirati pojedinačne aktivnosti mreže izrazito heterogenih tijela za postavljanje standarda?
- Bit će bitno omogućiti svim zainteresiranim dionicima da sudjeluju u procesu standardizacije prema IoT-u i da izraze svoje zahtjeve i brige. Kako se to može postići?

- Mnogi standardizacijski napori pokušavaju definirati jedinstveni prikaz podataka i protokol za IoT.
- Ovo je pravi smjer iako su neki pristupi ograničeni svojim opsegom domena primjene i tehnologija koje se koriste.
- Kako pretvoriti IoT u Web of Things?
- Postoji mnogo različitih razina protokola, ali oni koji se najizravnije odnose na poslovna i društvena pitanja su oni najbliži vrhu, takozvani aplikacijski protokoli

- BITXML, data format defined by BITX Inc.
- CBRN, format for Chemical, Biological, Radiological and Nuclear data
- CAP, Common Alerting Protocol, of EXDL
- EDDL, Electronic Device Description Language
- EEML, Extended Environments Markup Language from Pachube
- EXDL, Emergency Data Exchange Language of OASIS
- FDT, Field Device Tool
- IRIG, Inter-Range Instrumentation Group
- TransducerML, Transducer Markup Language of OGC
- WMMP, Wireless Machine Management Protocol of China Mobile

- MDMP, M2M protocol of China Telecom
- M2MXML, Machine-to-Machine XML
- NGTP, Next-Generation Telematics Protocol
- oBIX, open Building Information eXchange
- OMA SyncML, Open Mobile Alliance Synchronization Markup Language
- oMIX, open Machine Information eXchange
- OPC, OLE for Process Control
- PML, Physical Markup Language of EPCglobal
- SensorML, Sensor XML of OGC
- TEDS/IEEE 1451, transducer electronic data sheets of IEEE

# Pristupi za jedinstven podatkovni standard



- Postoje mnogi napor i da se stvori jedinstveni, međusegmentni, sveobuhvatni standard formata podataka za WoT
- Zbog razlika u različitim domenama ovo je veliki tehnološki izazov ili čak nemoguća misija
- Vjerojatno je realnije stvoriti standarde interoperabilnosti za integraciju WoT sustava.

- Kada WoT aplikacije prevladaju u budućnosti, globalno jedinstvena identifikacija objekata postat će ozbiljan problem.
- Identifikacija objekta može obuhvatiti imenovanje, adresiranje ili oboje, sredstva ili uređaja.
- Web – URI
- U IoT-u, slično Internetu i webu, objekti moraju imati zajedničke sheme imenovanja i adresiranja te također usluge otkrivanja kako bi se omogućila globalna referenca i pristup.

## Jedinstvena identifikacija objekata (2)

- ubiquitous ID (uID) okvir – razvijen u Japanu - 128-bitni sustav identifikatora fiksne duljine
- U području RFID-ova, EPCglobal [51] je promovirao usvajanje i standardizaciju elektroničkog koda proizvoda (EPC), koji se koristi za jedinstvenu identifikaciju RFID oznaka. Temelji se na URI modelu. - EPC (priznat u Sjedinjenim Državama i Europi) je konkurenčki standard uID-a (koristi se u Japanu)
- The international mobile equipment identity (IMEI) – jedinstvena identifikacija mobilnih uređaja - IMEI se formira skupom znamenki koje predstavljaju proizvođača, samu jedinicu i softver instaliran na njoj.

- OID je dobar kandidat za identifikaciju IoT objekata s obzirom na to da je to zrela shema koju podržavaju i ISO i ITU.
- Međutim, malo je složen za korištenje (budući da je dio ASN.1 koji uključuje procese registracije itd.) u usporedbi s drugim shemama kao što su UUID, EPC ili uID.
- S obzirom na to da EPC i uID nisu međusobno kompatibilni, UUID je široko prihvaćena shema koja se koristi u distribuiranim okruženjima uključujući IoT.

- UUID (universal unique identifier)
- Standard za identifikatore korišten kod razvoja softvera, standardiziran od strane Open Software Foundation (kasnije nazvan Open Group) kao dio Distributed Computing Environment.
- Namjera UUID-ova je omogućiti distribuiranim sustavima jedinstvenu identifikaciju informacija bez značajne središnje koordinacije.
- Koriste se u Microsoft-ovom GUID-ovima

- ANSI/ISA-95 - međunarodni standard za razvoj automatiziranog sučelja između poduzeća i sustava upravljanja
- Ciljevi ISA-95 su osigurati dosljednu terminologiju koja je temelj za komunikaciju dobavljača i proizvođača, osigurati dosljedne informacijske modele i osigurati dosljedan operativni model kao temelj za pojašnjavanje funkcionalnosti aplikacije i načina na koji se informacije trebaju koristiti.

- Constrained Application Protocol (CoAP)
- Message Queuing Telemetry Transport (MQTT)
- WiFi
- ZigBee
- Bluetooth
- Extensible Messaging and Presence Protocol (XMPP)
- Data-Distribution Service (DDS)
- Advanced Message Queuing Protocol (AMQP)
- Lightweight M2M (LwM2M)

- Osmišljen od strane radne skupine IETF-a za ograničena RESTful okruženja i pokrenut 2013., Constrained Application Protocol (CoAP) dizajniran je za prevodenje HTTP modela tako da se može koristiti u restriktivnim uređajima i mrežnim okruženjima.
- Bazira se na User Datagram Protocol (UDP) za uspostavljanje sigurne komunikacije između krajnjih točaka
- CoAP u potpunosti zadovoljava potrebe iznimno laganog protokola kako bi zadovoljio zahtjeve uređaja koji rade na baterije ili uređaja niske potrošnje.

- Najviše korišten standard u IoT-u
- Lagani protokol za razmjenu poruka vrste objave/preplate (pub/sub).
- Dizajniran za uređaje koji se napajaju baterijama, MQTT-ova arhitektura je jednostavna i lagana, pružajući nisku potrošnju energije za uređaje.
- Radeći na vrhu TCP/IP protokola, posebno je dizajniran za nepouzdane komunikacijske mreže kako bi se odgovorilo na problem sve većeg broja malih jeftinih objekata male snage koji su se pojavili u mreži posljednjih godina.

- Mreže temeljene na ZigBee karakterizira niska potrošnja energije, niska propusnost (do 250 kbps) i raspon povezivosti od 100 metara između čvorova.
- Tipične primjene uključuju senzorske mreže, osobne mreže (WPAN), kućnu automatizaciju, alarmne sustave i sustave za nadzor.
- ZigBee je razvijen kao standard za samokonfigurirajuće radio mreže kratkog dometa, namijenjen za korištenje u telemetrijskim sustavima, za komunikaciju između različitih tipova senzora, nadzornih uređaja, kao i za bežično očitavanje rezultata mjerena s mjerača energije i topline.

- Bluetooth je tehnologija koja omogućuje bežično povezivanje različitih elektroničkih uređaja, poput telefona, tipkovnice, računala, prijenosnog računala, miša, dlanovnika, pisača, slušalica ili zvučnika i još mnogo toga.
- Standard koristi radio valove u ISM frekvencijskom pojasu od 2,4 GHz

# Extensible Messaging and Presence Protocol (XMPP)



- Komunikacijski IoT protokol za međuprogram orijentiran na poruke
- Temelji se na XML jeziku



# Servisno orijentirana arhitektura za Internet stvari

Darko Andročec

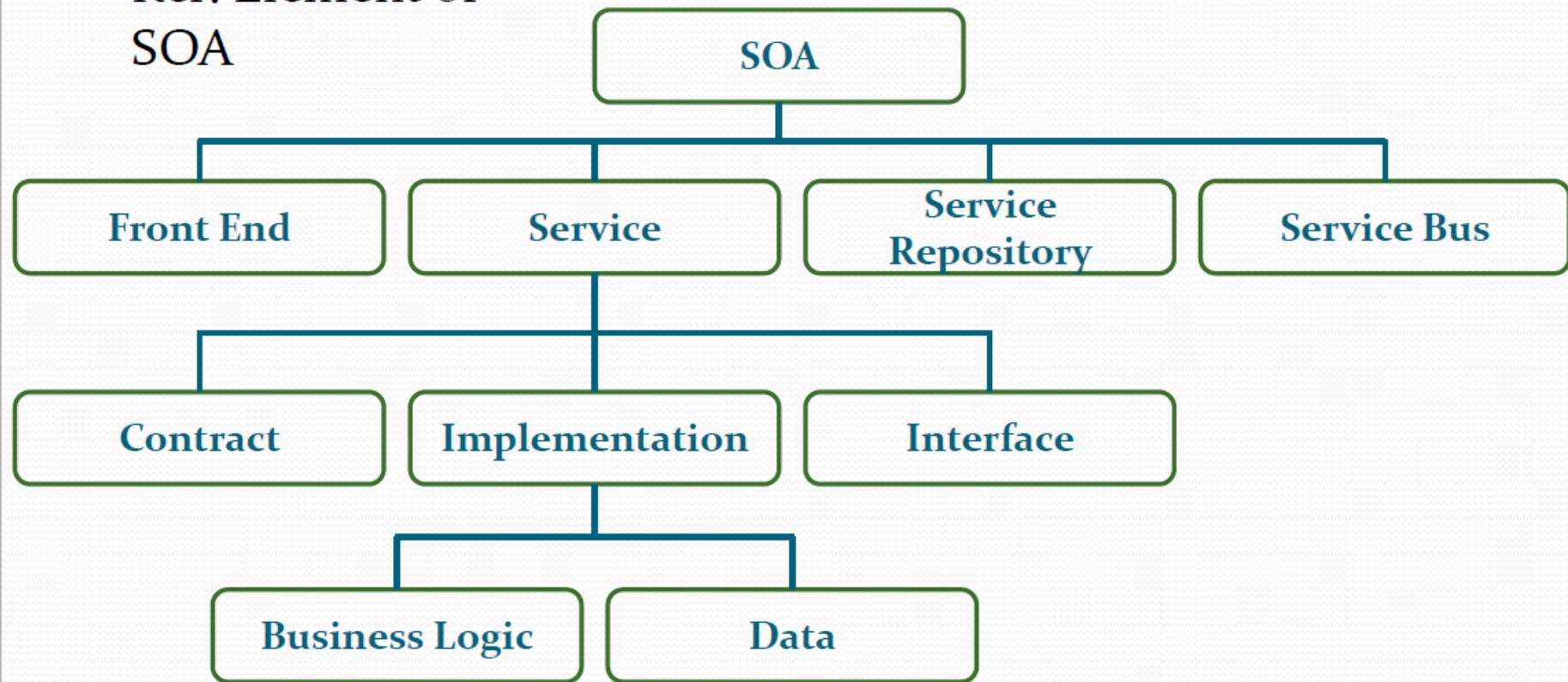
## Definicija SOA

- Sommerville - Servisno orijentirane arhitekture (SOA) način su razvoja distribuiranih sustava gdje su komponente sustava samostalne usluge koje se izvode na geografski distribuiranim računalima.
- OASIS - Paradigma za organiziranje i korištenje distribuiranih mogućnosti koje mogu biti pod kontrolom različitih vlasničkih domena. Pruža jednoobrazno sredstvo za ponudu, otkrivanje, interakciju i korištenje mogućnosti za postizanje željenih učinaka u skladu s mjerljivim preuvjetima i očekivanjima.

- Servis - labavo povezana softverska komponenta za višekratnu upotrebu koja sadrži diskretnu funkcionalnost, koja se može distribuirati i kojoj se može programski pristupiti.
- Konektori – poruke, metapodaci (opis servisa, sučelje servisa, semantički metapodaci)

# Sve komponente SOA-e

Ref: Element of SOA



- Tehnologija
- Middleware
- Implementacija SOA-e

- Labavo povezivanje - servisi održavaju odnos koji smanjuje ovisnost i samo održavaju svijest jedan o drugome
- Ugovor o usluzi - servisi se pridržavaju komunikacijskog sporazuma kako je zajednički definirano jednim ili više dokumenata s opisom usluge
- Apstrakcija usluge - Osim onoga što je opisano u ugovoru o usluzi, servisi skrivaju logiku od vanjskog svijeta
- Ponovna upotreba usluge - Logika je podijeljena na servise s namjerom promicanja ponovne upotrebe

- Kompozitivnost servisa - Zbirke servisa mogu se koordinirati i sastaviti u složene servise
- Autonomija usluge – Servisi imaju kontrolu nad logikom koju enkapsuliraju
- Optimizacija servisa – visokokvalitetne usluge općenito se smatraju boljim od onih niske kvalitete
- Mogućnost otkrivanja usluge - usluge su osmišljene tako da budu opisane prema van kako bi se mogle pronaći i procijeniti putem dostupnih mehanizama otkrivanja

- Smanjeni troškovi
- Povećanje produktivnosti zaposlenika
- Izrađene za integraciju i partnerstva
- Agilnost – izgrađena za promjene

- Kririranje skalabilnih sustava koji mogu evoluirati
- Bolje upravljanje kompleksnim sustavima
- Neovisnost o pojedinoj platformi
- Labavo povezivanje omogućuje fleksibilnost

- Upravljanje servisima – premala pažnja upravljanju servisa može prouzročiti probleme s performansama i pouzdanošću
- Pružanje odgovarajuće sigurnosti za pojedine uloge
- Osiguranje interoperabilnosti servisa

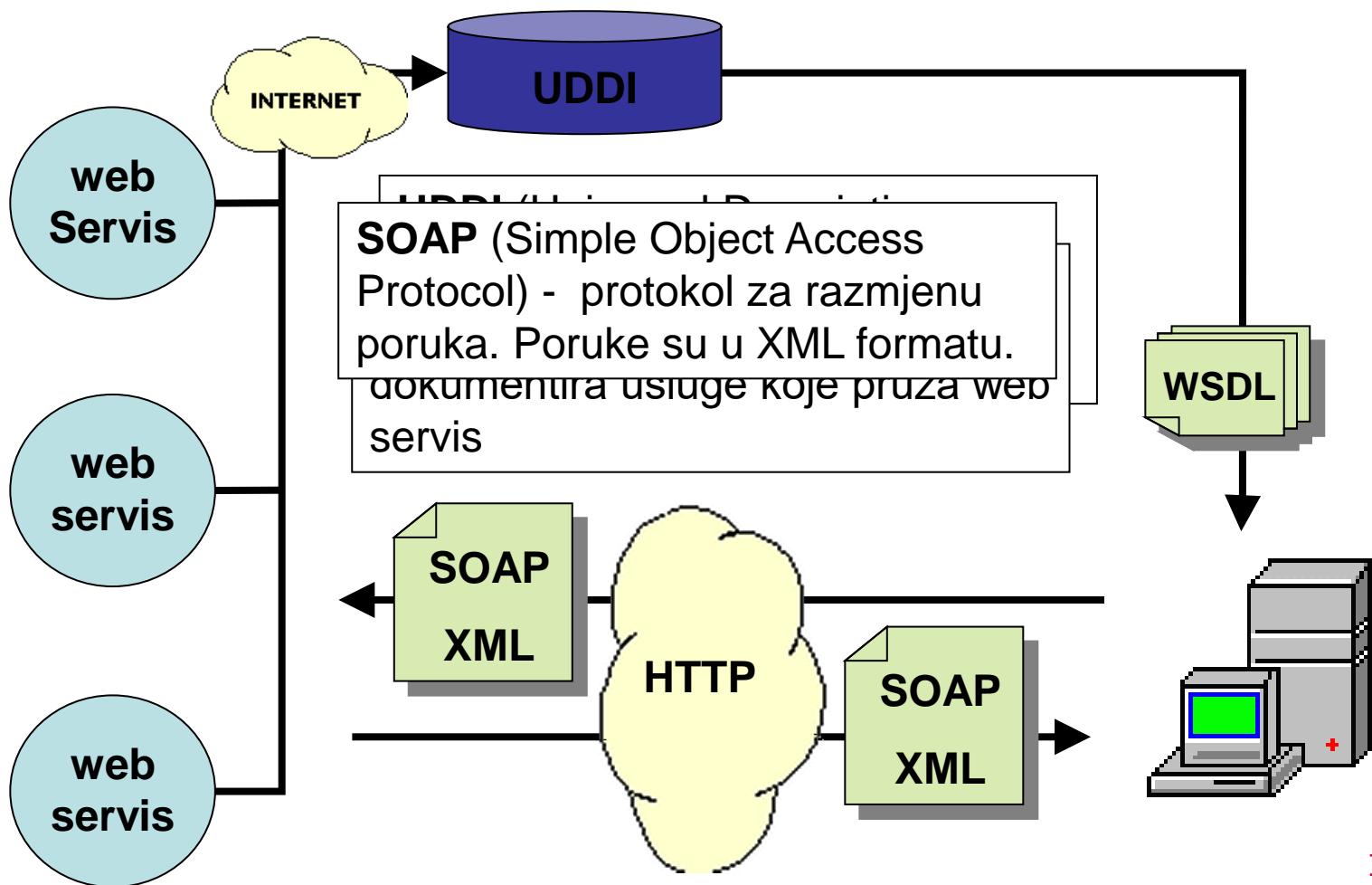
## Kada ne koristiti SOA pristup?

- U homogenim IT okruženjima
- Kada je kritična performansa u realnom vremenu
- Kada je poželjno čvrsto povezivanje (tight coupling)
- Kada se stvari ne mijenjaju

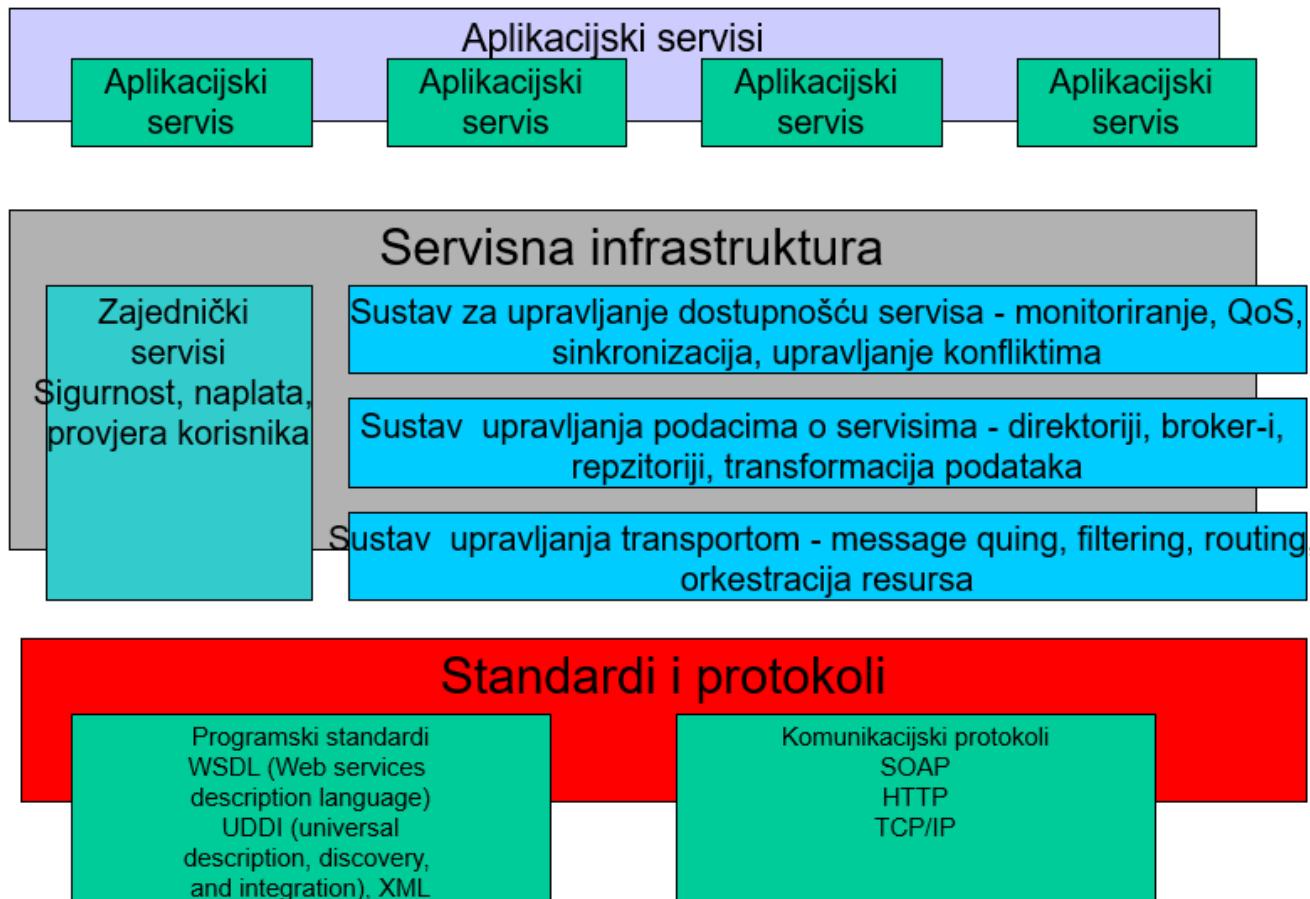
- omogućuje razmjenu podataka između aplikacija i web portala
- podaci se ne prepisuju, nego se dohvaćaju direktno s mjesta nastanka
- umjesto zamornog traženja, korisnik sve informacije o traženom sadržaju dobiva na jednom mjestu

- softverska aplikacija koja nije vezana za operacijski sustav ili programski jezik
- distribuirana okruženja - Internet ili intranet
- standardizirani sustav za razmjenu poruka
- razmjena podataka
- obavljanje poslovnih transakcija

## Web servisi (2)



## Arhitektura Web servisa

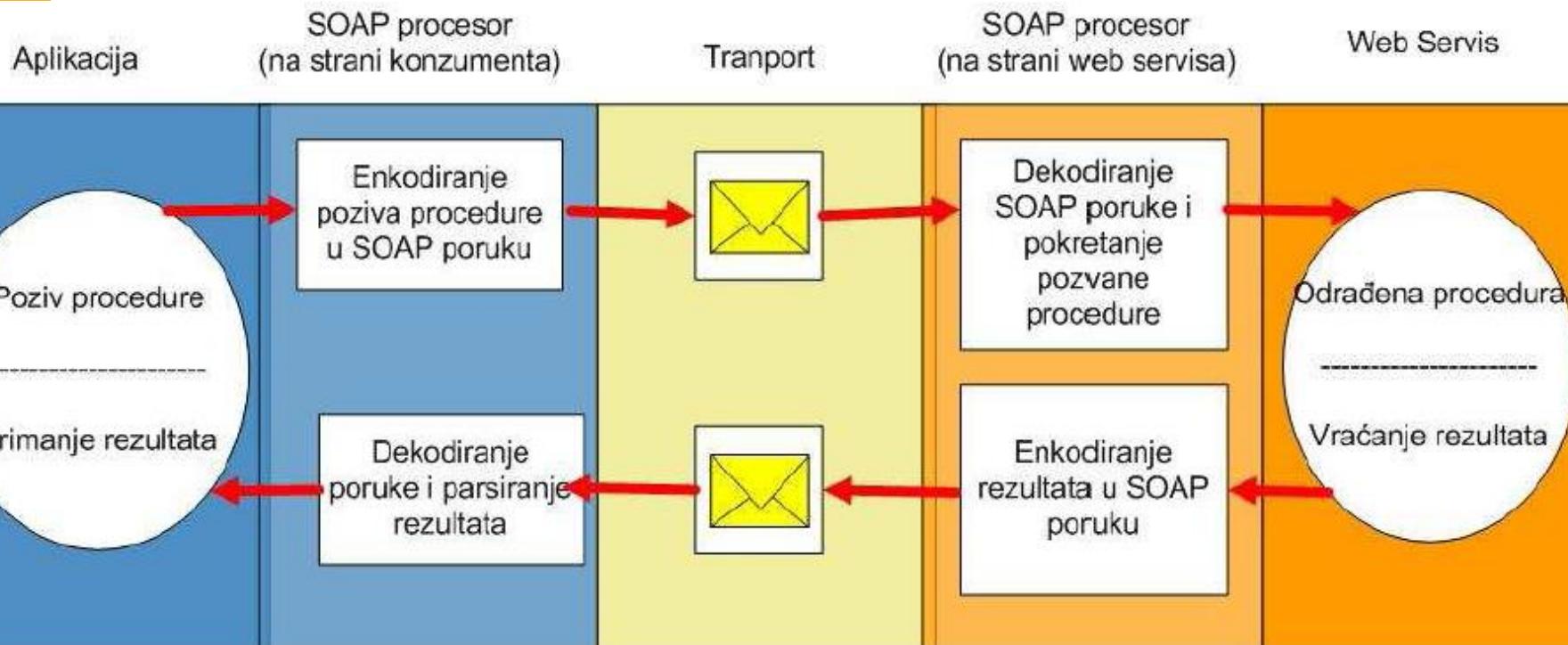


## Grupa protokola

- XML (eXtensible Markup Language)
- SOAP (Simple Object Access Protocol)
- WSDL (Web Service Description Languages)
- UDDI (Universal Description, Discovery and Integration)

- W3C XML bazirani komunikacijski protokol za razmjenu poruka između aplikacija
- ima sve prednosti XML-a
- komunikacija između različitih aplikacija dostupnih putem Internet protokola

# SOAP (2)



# Sintaksa SOAP-a

- XML dokumenti strukturirani po dodatnom skupu pravila
- SOAP Envelope element - root element
- SOAP Body element - nositelj poruke
- SOAP Header element - dodatni podaci
- SOAP Fault element - poruke o greškama



- standard za dokumentiranje usluga koje pruža web servis
- notacija je bazirana na standardnoj XML shemi
- u jednoj XML datoteci definira sve što je potrebno za pisanje korisničkog softvera koji koristi usluge Web servisa

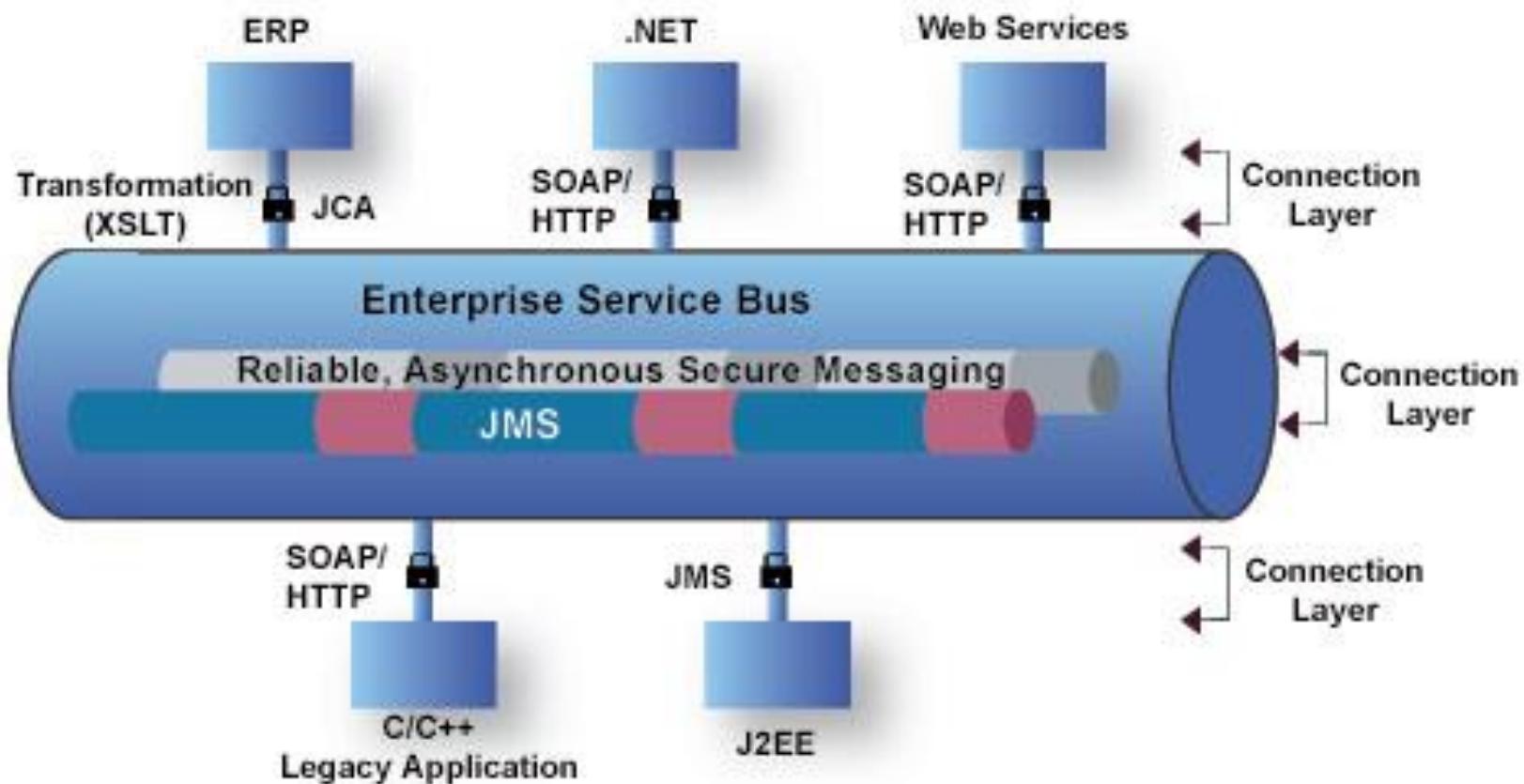
- implementacija UDDI specifikacije je UDDI Business Registry
- bijele stranice - opisuju pružatelja usluge
- žute stranice - industrijske kategorije
- zelene stranice - opisuju sučelje servisa (WSDL)

- **WS-I** (Web Services Interoperability Organization) - <http://www.ws-i.org/> - sada dio OASIS-a
- promicanje novih standarda potrebnih da koncept Web servisa zaživi u potpunosti
- **WS-\* specifikacije** (WS-Security, WS-Policy, WS-SecurityPolicy, WS-Trust, WS-SecureConversation, WS-Privacy, itd.)

## ESB za integraciju (1)

- Enterprise Service Bus
- Skup kontejnera servisa koji objedinjavaju različite vrste IT resursa
- Kontejneri su međusobno povezani sustavom razmjene poruka
- Kontejneri prilagođavaju IT resurse prema standardiziranom servisnom modelu (komunikacija XML porukama)

- Središnja administracija i nadzor distribuiranog sustava upravljanjem i razmjenom poruka (message transformation services, message routing services)
- Omogućuje ad-hoc promjene u sustavu, bez potrebe za narušavanjem funkcionalnosti sustava
- Middleware
- Povezivanje podsustava i aplikacija
- Posredovanje – usmjerenje, modeli interakcije, upravljanje verzijama
- Upravljanje i nadzor – upravljanje porukama, transakcijama, sigurnošću, QoS-om...



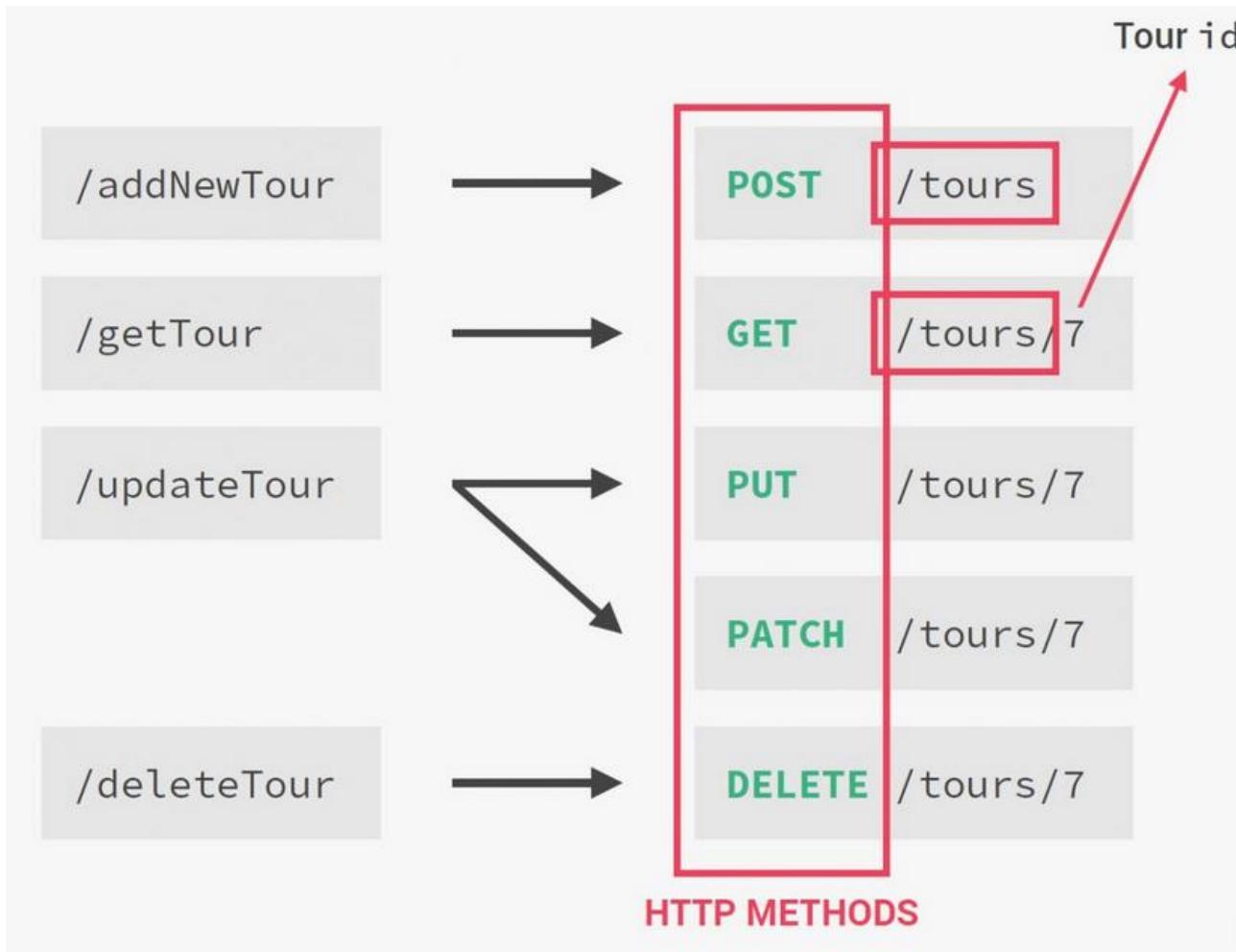
## WS-BPEL (1)

- Specifikacija OASIS-a
- WS-BPEL (engl. Web Services – Business Process Execution Language)
- Definira jezik za izgradnju poslovnih procesa zasnovanih na web servisima
- WS-BPEL (tj. njezina prva verzija BPEL4WS) nastala je 2002. godine kao rezultat zajedničkog rada tvrtki IBM, Microsoft i BEA
- U razvoj jezika priključile su se i neke druge softverske tvrtke kao što su SAP i Siebel Systems.
- Specifikacija je u 2007. predana organizaciji OASIS kako bi postala službena, otvorena norma

- WS-BPEL koristi već postojeće mehanizme specifikacije web servisa – poglavito normu XML te normu za opis Usluga weba WSDL.
- Glavna ideja jest ta da se stvori jedna upravljačka usluga – tzv. poslužitelj BPEL procesa – koja na osnovi danog XML-ovskog predloška ima odgovornost provođenja poslovnog procesa.
- Upravljačka usluga naziva se BPEL-ovskim procesorom te ima svoj vlastiti WSDL-ovski opis te je i sama po svim karakteristikama jedan klasični web servis.

- REST = Representational State Transfer
- Jednostavniji, više integrirani sa HTTP-om
- Ne zahtijevaju XML poruke niti WSDL opise servisa
- Danas se češće koristi nego SOAP
- Podaci se vraćaju najčešće u JSON formatu – rjeđe se koristi XML i YAML
- Resursi imaju URL ili URI kojima se identificiraju
- Svaki poziv čini jednu akciju (kreiranje, čitanje, mijenjanje ili brisanje podataka)
- Isti URL se koristi za sve operacije, ali se mijenja HTTP metoda koja definira vrstu operacije

# Primjer REST servisa



- Mogućnost privremenog spremanja (Cacheable)
- Jedinstveno sučelje (Uniform interface URI)
- Izričito korištenje HTTP metoda
- Prijenos XML-om ili JSON-om
- Nepostojanje stanja („stateless”) – poslužitelj ne pamti nikakve podatke o klijentu, klijent sa svakim zahtjevom mora slati sve potrebne informacije za razumijevanje i obradu tog zahtjeva

# Prednosti RESTfull servisa

- Jednostavnost
- Fleksibilnost formata vraćenih podataka
- Korištenje postojeće mrežne infrastrukture
- Brzo savladavanje tehnike

- Ključna apstrakcija informacija u REST-u je resurs
- Bilo koja informacija koja se može imenovati može biti resurs – dokument ili slika, privremeni servis, skup drugih resursa ili stvarni objekt (npr. osoba)
- Stanje resursa u određeno vrijeme se naziva reprezentacijom resursa
- Reprezentacija resursa sastoji se od podataka, metapodataka koji opisuju podatke i hipermedijskih linkova koji pomažu klijentu pri prijelazu na sljedeće željeno stanje

- REST API sastoji se od skupa međusobno povezanih resursa
- Resursima se pristupa putem URI-ja (Uniform Resource Identifiers)
- API (engl. application programming interface) je skup pravila koja definiraju kako se aplikacije ili uređaji mogu međusobno povezivati i komunicirati.
- REST API-ji su uobičajena metoda za povezivanje komponenti i aplikacija u arhitekturi mikroservisa.

- Mikroservisna se arhitektura sastoji od malih servisa, zvanih mikroservisi (dalje u tekstu: servisi), autonomnih programskih komponenti koje pružaju usluge drugim servisima i s kojima surađuju.
- Cilj razvoja servisa u mikroservisnoj arhitekturi je da u konačnici servis bude što manji, optimiziran i fokusiran isključivo na svoju domenu.
- Servisi moraju biti međusobno neovisni
- Servisi koriste aplikacijsko programsko sučelje (eng. application programming interface - API) za komuniciranje i surađivanje s drugim servisima putem mreže.
- Svaki servis pruža svoje sučelje drugim servisima pa odabir tehnologije sučelja nije trivijalan zadatak.

# Razlika između SOA-e i mikroservisa

SOA	Mikroservisna arhitektura
Maksimiziranje ponovnog korištenje servisa	Fokusira se razdvajanje
Sustavna promjena zahtjeva izmjenu monolita	Sustavna promjena znači dodavanje novog servisa
DevOps i Continuous Delivery postaju popularni ali ne nužni	Veliki fokus na DevOps i Continuous Delivery
Fokus na ponovnom korištenju poslovne funkcionalnosti	Veća važnost na koncept ograničenog konteksta
Za komunikaciju se koristi Enterprise Service Bus (ESB)	Za komunikaciju se koriste šturi i jednostavnii sustavi poruka
Podržava nekoliko protokola poruka	Koristi jednostavne protokole poput HTTP-a, REST-a ili RPC-a
Koristi zajedničku platformu za razmještanje svih servisa	Aplikacijski poslužitelji se ne koriste, normalno je da se koriste platforme u oblaku
Korištenje kontejnera poput Docker-a nije toliko popularno	Kontejneri rade vrlo dobro s servisima
SOA servisi dijele spremište podatka	Svaki servis može imati svoje spremište podataka
Zajedničko upravljanje i standardi	Opušteno upravljanje, s većim naglaskom na suradnju timova i slobodu izbora

- Slabo povezani servisi ne znaju puno jedni o drugima, čak i kada neposredno surađuju.
- Cilj je visoke kohezije (eng. high cohesion) grupiranje logike programa u povezane cjeline.
- Svaka cjelina sadrži logiku i ponašanje koje je usko vezano za njezin kontekst, dok je sve ostalo, što nije direktno vezano za taj kontekst, u drugoj cjelini.
- Uzevši u obzir tehnologije koje se koriste u mikroservisnoj arhitekturi, tehnologije integracije imaju najveću važnost.
- Ispravnim odabirom tehnologije integracije ona se neće morati mijenjati za vrijeme razvoja aplikacije, čime se izbjegava teška izmjena svih servisa zbog takve promjene.

## WebSocket protokol (1)

- WebSocket protokol je po mnogima najbolja tehnologija i komunikacijska tehnika dostupna za korištenje u svrhu izgradnje Web aplikacija u realnom vremenu.
- Pruža perzistentnu, asinkronu, dvosmjernu komunikaciju (eng. full duplex) preko jedne TCP veze kojom je moguće slati neograničen broj zahtjeva tako dugo dok je veza otvorena i to sa vrlo malim troškom u resursima aplikacije.
- Standardiziran je od strane W3C konzorcija 2011. godine sa pripadnim WebSocket okvirom za razvoj (eng. API), a danas ga podržava velika većina modernih web preglednika

- Veza koja se otvara WebSocket protokolom je takve prirode da dopušta i klijentu i poslužitelju da, neovisno jedan o drugome, šalju poruke jedan drugome, čak i u isto vrijeme.
- Također, zbog malog trošenja resursa poput radne memorije, poslužitelj može u isto vrijeme imati otvoren velik broj veza prema više klijenata koristeći WebSocket protokol što omogućuje izradu skalabilnih Web aplikacija u realnom vremenu, pogotovo ako očekujemo da aplikaciju koristi velik broj korisnika u isto vrijeme.

- Web servisi ključni su za razvoj IoT-a
- Pogotovo se koriste u WoT, čiji su zapravo tehnološki temelj



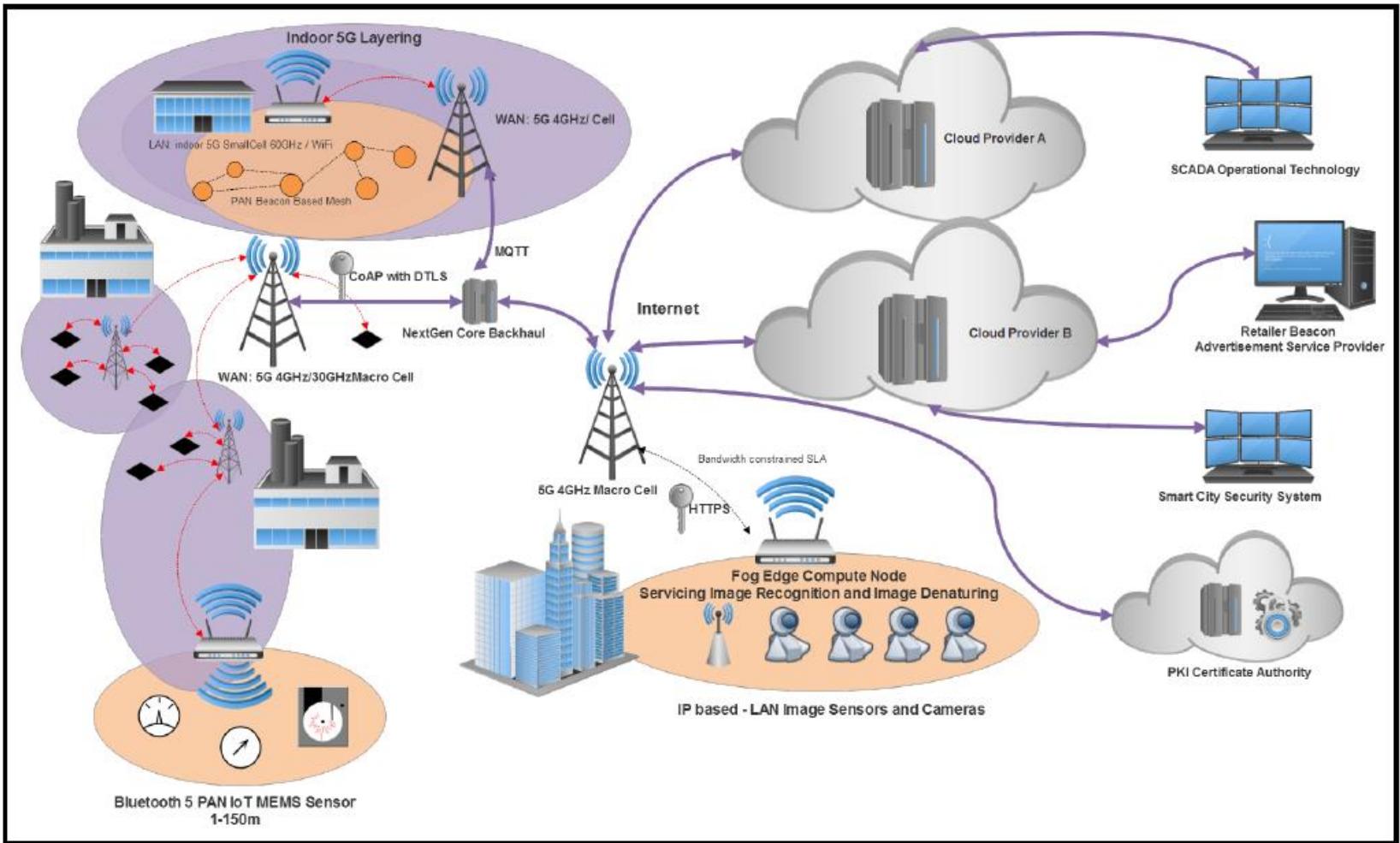
# Arhitektura weba stvari

Darko Andročec

- Senzori
- Senzorski komunikacijski sustavi
- Lokalne mreže
- Agregatori, usmjerivači, gatewayi
- WAN
- Oblak
- Analiza podataka
- Sigurnost

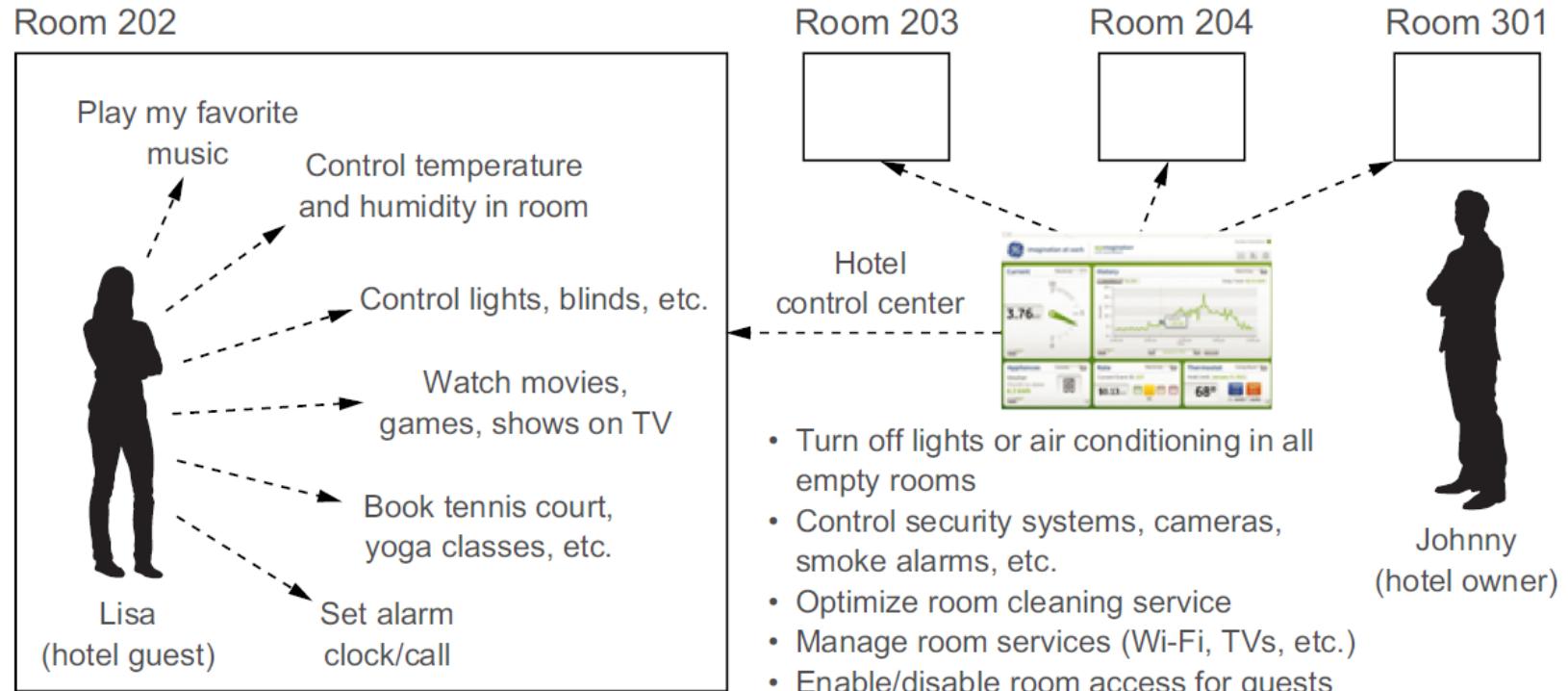
- M2M - To je opći koncept koji uključuje izravnu komunikaciju autonomnog uređaja s drugim autonomnim uređajem. Autonomno se odnosi na sposobnost čvora da instancira i komunicira informacije s drugim čvorom bez ljudske intervencije.
- IoT - IoT sustavi mogu uključivati neke M2M čvorove (kao što je Bluetooth mreža koja koristi ne-IP komunikaciju), ali prikupljaju podatke na rubnom usmjerivaču ili pristupniku.
- Činjenica da ima metodu povezivanja s internetskom strukturu je ono što definira IoT.

- Obuhvaća mnoge tehnologije



- Kreiranje jedinstvenog i globalnog ekosustava stvari koje komuniciraju jedna s drugom je danas gotovo nemoguće
- Nema jedinstvenog aplikacijskog protokola za IoT
- Umjesto da se izmišlja novi protokol, ideja se da se koriste web protokoli -> stvaranje web stvari
- Web stvari omogućuje da su podaci i servisi stvari dostupniji većem skupu web programera

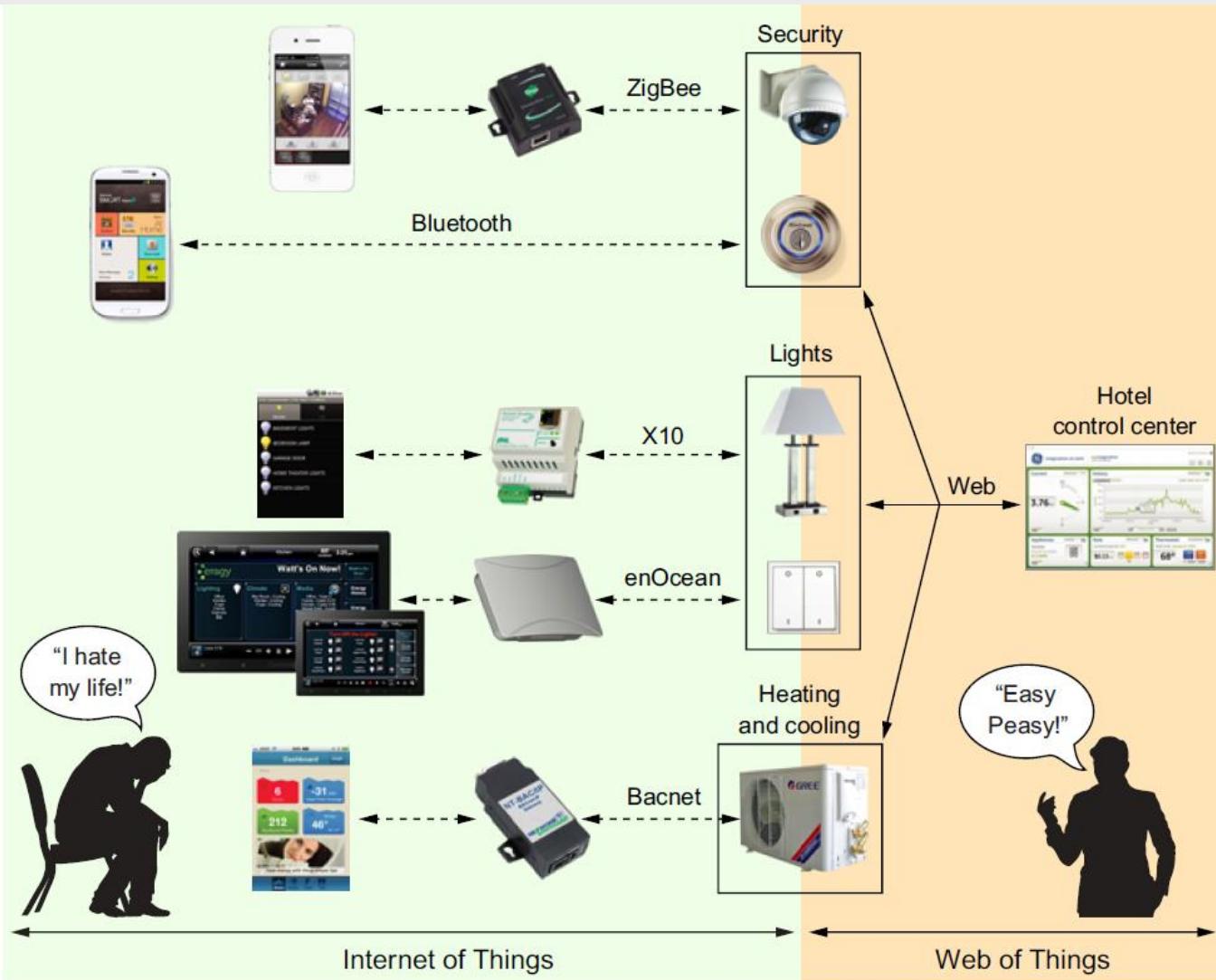
# Scenarij weba stvari – povezani hotel (1)



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

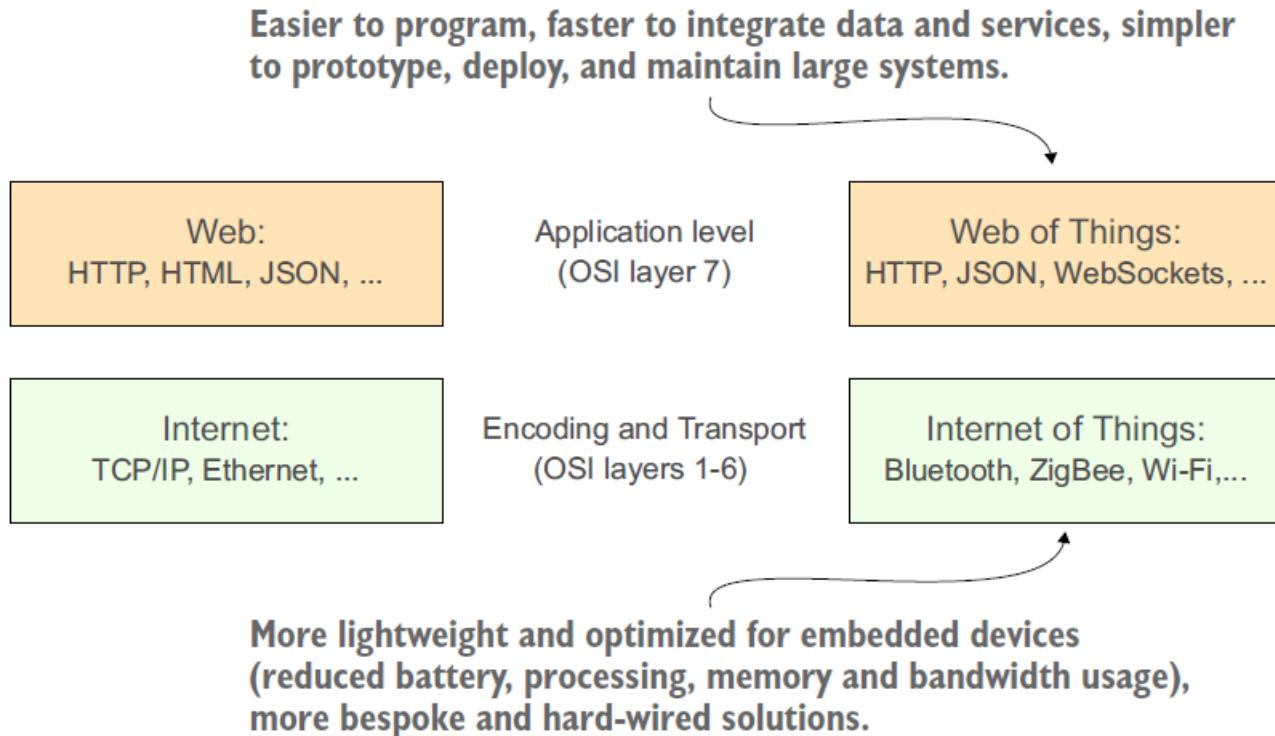
- Johnny bi želio digitalno povezati uređaje u svim sobama svog hotela.
- Prvo, gosti bi mogli imati pristup raznim uslugama od kontrole svoje sobe (svjetla, klimatizacija, zabava, itd.), do rezerviranja hotelskih objekata, do naručivanja hrane i pića—sve to na svojim mobilnim telefonima.
- Drugo, ovaj bi sustav omogućio Johnnyju da koordinira i optimizira sve aspekte svog hotela na centraliziran i učinkovit način, bez potrebe za korištenjem raznih zasebnih aplikacija i alata.

# Stotine nekompatibilnih IoT protokola



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

# Web stvari (1)



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Web of Things bavi se samo najvišim OSI slojem (7), koji upravlja aplikacijama, uslugama i podacima.
- Rad s tako visokom razinom apstrakcije omogućuje povezivanje podataka i usluga s mnogih uređaja bez obzira na stvarne transportne protokole koje koriste.
- Nasuprot tome, Internet of Things ne zagovara jedan protokol na razini aplikacije i obično se fokusira na niže slojeve OSI skupa.

## Prednosti web stvari

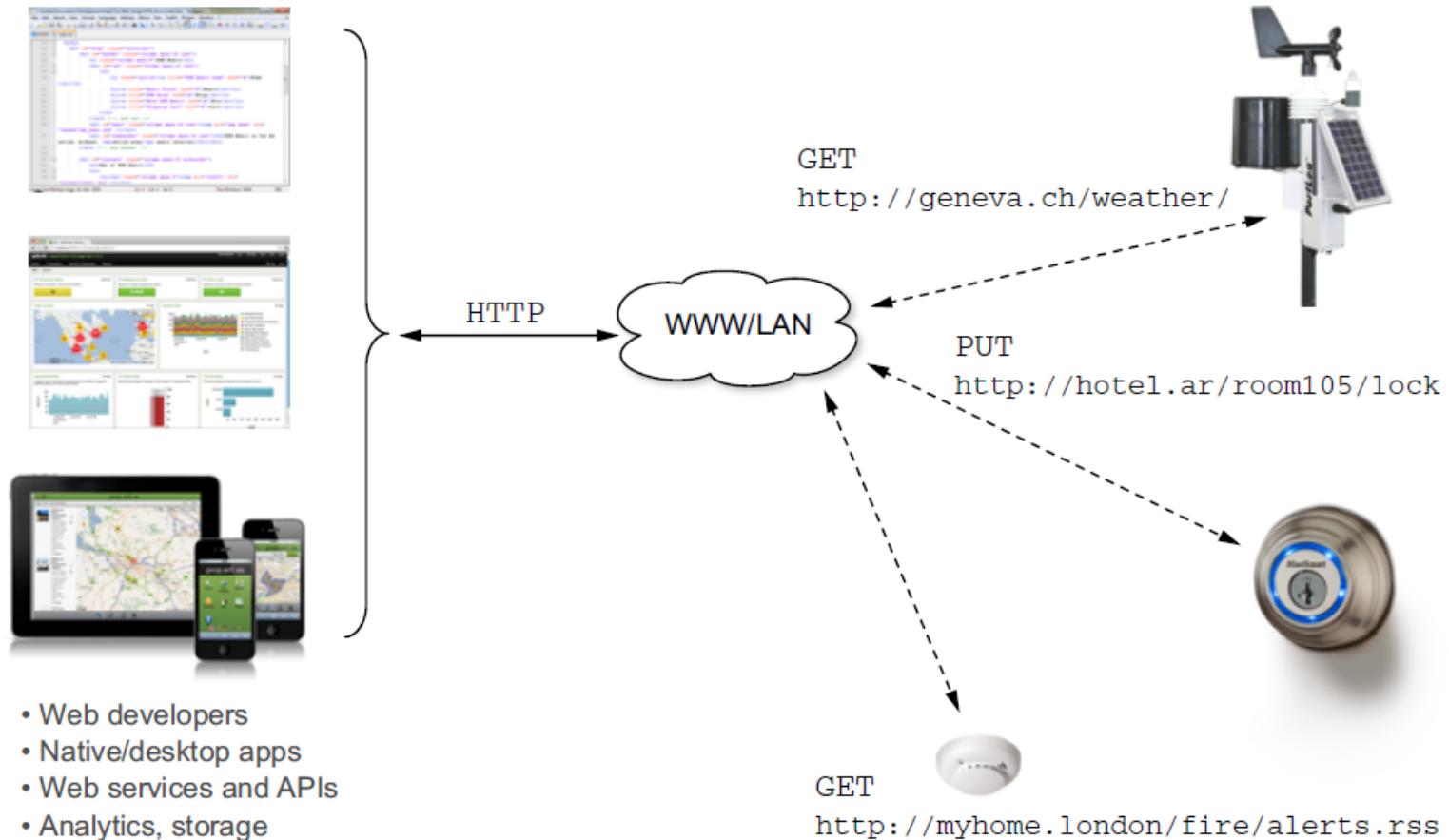
- Apstrahiranje složenosti i raznolikosti protokola niže razine iza jednostavnog modela weba nudi mnoge prednosti.
- Kao što je web postao globalna integracijska platforma za distribuirane aplikacije preko interneta, web stvari olakšava integraciju svih vrsta uređaja i aplikacija koje su u interakciji s njima.
- Drugim riječima, skrivajući složenost i razlike između različitih transportnih protokola koji se koriste u IoT-u, web stvari omogućuje programerima da se usredotoče na logiku svojih aplikacija bez potrebe da se zamaraju time kako ovaj ili onaj protokol ili uređaj zapravo funkcioniра.

- U webu stvari, uređaji i njihove usluge potpuno su integrirani u web jer koriste iste standarde i tehnike kao i tradicionalne web stranice.
- To znači da možete pisati aplikacije koje komuniciraju s ugrađenim uređajima na potpuno isti način kao što biste komunicirali s bilo kojom drugom web uslugom koja koristi web API-je — posebno RESTful arhitekturom.
- Korištenjem svih ovih standarda za scenarije Interneta stvari, omogućujemo izgradnju novih vrsta interaktivnih aplikacija i osiguravamo da se uređaji mogu integrirati s modernim web aplikacijama i uslugama uz minimalan napor.



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Za razliku od mnogih protokola i standarda koji postoje u Internetu stvari, programski model iza Weba stvari znatno je jednostavniji za naučiti i koristiti.
- Ovo je posebno zanimljivo jer svakome s osnovnim vještinama web programiranja omogućuje izradu web stranica i aplikacija, ne samo oko multimedijskog sadržaja, već i s podacima u stvarnom vremenu iz fizičkog svijeta.



- U nekim slučajevima ima smisla da stvari imaju javni URL i da budu otvoreno dostupne putem weba—na primjer, senzori za promet ili onečišćenje u gradu kojim upravljaju javne vlasti.
- U ovom slučaju, uređaji se također mogu indeksirati i indeksirati u web tražilicima poput bilo koje druge web stranice i omogućuju korisnicima da doslovno guglaju fizički svijet ili označe URL pametnog objekta i dijele ga s prijateljima.
- Objekti povezani s webom također mogu postati aktivni i sudjelovati na webu baš kao i drugi korisnici objavljuvanjem vlastitih blogova ili međusobnom komunikacijom koristeći API-je usluga kao što je Twitter.

- Koristeći usluge kao što je IFTTT, korisnici mogu stvoriti mala, logična pravila koja miješaju uređaje iz stvarnog svijeta kao što su senzori u njihovom domu s virtualnim uslugama u oblaku; na primjer, SMS pristupnik ili usluga vremenske prognoze.
- Takve se aplikacije nazivaju fizičkim kombinacijama (physical mashups).

# Bežični senzorski čvorovi



Mica Mote (~2004):

CPU: Atmel ATmega 128 @ 4 MHz  
Flash: 128 KB RAM + 512 KB Flash  
Radio: 868/916 MHz



Sun SPOT (~2010):

CPU: ARM ARM920T @ 180 MHz  
Flash: 512 KB RAM + 4 MB Flash  
Radio: 2.4 MHz IEEE 802.15.4



Intel Edison (~2014):

CPU: Atom dual core @ 500 MHz  
Flash: 1 GB RAM + 4 GB Flash  
Radio: Wi-Fi + Bluetooth

Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Jasno je da većina WSN uređaja nije dizajnirana za javnost.
- Te su platforme uglavnom programirali stručnjaci.
- Iako su web protokoli teži (opširniji) od optimiziranih protokola koji se koriste na ugrađenim uređajima, došlo je do puno napretka u optimiziranim HTTP bibliotekama koje se izvode na ograničenim uređajima.

- Osim toga, uređaji postaju sve moćniji i mnogi dolaze s Wi-Fi vezom.
- Mogućnost interakcije s ugrađenim senzorima korištenjem standardnih web protokola čini prikupljanje, pohranjivanje i analizu podataka iz heterogenih senzora mnogo jednostavnijim.
- Doista, integracija podataka u nekoliko usluga u oblaku puno je brža zahvaljujući jednostavnosti i sveprisutnosti REST API-ja.

- Još jedan zanimljiv slučaj upotrebe za IoT je izgradnja sićušnih senzora koje ljudi mogu nositi kako bi pasivno pratili svoje dnevne aktivnosti ili čak tjelesne čimbenike kao što su otkucaji srca ili kemikalije u krvi ili znoju.
- Monitori otkucaja srca odavno su komercijalizirani za trkače na duge staze kako bi pratili i regulirali rad srca.
- Ovaj trend je doživio procvat u posljednjih nekoliko godina s mnogo više proizvoda u rasponu od uređaja za praćenje aktivnosti, do pametnih vaga povezanih s vašim telefonom koje vam pomažu kontrolirati težinu i tjelesnu masnoću, do pametnih pedala koje prate vaše vožnje i rade kao uređaji protiv krađe, do pametnih jastuka, pametne kutije za tablete, budilice i pametnih satova.

- Integracija nosivih i kvantificiranih vlastitih uređaja na webu, tako da su podaci izravno dostupni drugim uređajima i aplikacijama, znatno će olakšati razvoj novih klasa proširivih aplikacija za njegu starijih osoba, zdravlje i fitness, ili zabavu i sport.
- Također će osigurati da ne trebate zasebnu aplikaciju za svaku od njih (sa zanimljivim izazovima sigurnosti i privatnosti).

- Okruženje pametnog doma vjerojatno je simptomatično za (pre)veliki broj standarda i protokola koji postoje za povezivanje stvari s mrežama.
- Iako bi svi uređaji u vašem domu trebali međusobno komunicirati, ne mogu jer su ti protokoli nekompatibilni i na kraju imate više aplikacija i daljinskih upravljača nego ikad prije.
- Web stvari nudi alternativni pristup gdje su web jezici osnova za minimalni API koji bi uređaji trebali nuditi izravno ili neizravno putem pristupnika.

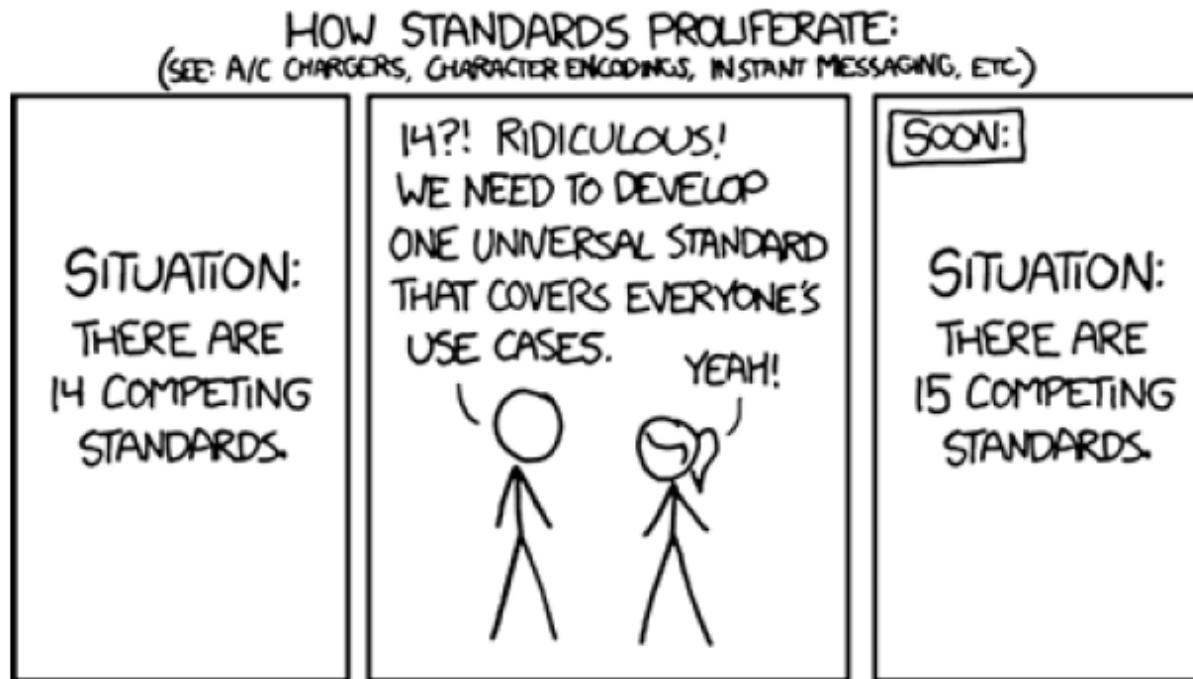
- Korištenje web standarda u kontekstu pametnih gradova posebno je zanimljivo jer oni uvelike olakšavaju dijeljenje podataka senzora s javnošću i programerima te olakšavaju korištenje podataka u stvarnom vremenu o prometu, zagađenju ili javnom prijevozu u njihovim vlastitim urbanim aplikacijama.

- IoT omogućava industriju 4.0
- Korištenje web standarda za međusobno povezivanje svih elemenata u poslovnom procesu, kao što su radni strojevi, poslovni softver, zaposlenici u raznim odjelima, proizvodi, kupci i dobavljači, predstavljat će značajnu promjenu u načinu na koji tvrtke posluju.
- Kada svi sudionici u tim procesima budu u mogućnosti automatski odlučiti kako najbolje izvršiti svoju dužnost na temelju podataka u stvarnom vremenu, nema sumnje da će se način na koji dizajniramo, proizvodimo i distribuiramo fizičke proizvode duboko promijeniti.

- Zamislite lanac opskrbe omogućen webom koji u stvarnom vremenu zna temperaturu vaših jagoda i može poslati upozorenja čim se uvjeti promijene ili čak automatski regulirati temperaturu kamiona, brodova i skladišta prema vrsti proizvoda koji se skladište i transportirani—sve te informacije dostupne putem web API-ja.
- Dijeljenje povijesnih podataka o uređajima koji koriste web standarde znatno će olakšati zajednički rad više aplikacija kroz cijeli životni ciklus proizvoda.
- To znači puno niže troškove integracije i visoku cjelovitost podataka u različitim sustavima koji će obrađivati i rukovati tim proizvodima.

- Mobilne aplikacije mogu dohvati podatke o CPG i FMCG proizvodima, komunicirati s njima kako bi priložili digitalni sadržaj i dijeliti informacije o njima na društvenim mrežama puno brže i jednostavnije putem weba.
- Kad bi svaki proizvod na svijetu imao vlastiti URL i web API, svakoj bi aplikaciji bilo lako prepoznati proizvod i pristupiti njegovim podacima bez puno napora pri integraciji.

# Problem IoT-a



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Jednostavnije programiranje
- Otvoreni i proširivi standardi
- Brzo i jednostavno postavljanje, održavanje i integracija
- Labavo povezivanje elemenata
- Poznati mehanizmi sigurnosti i privatnosti

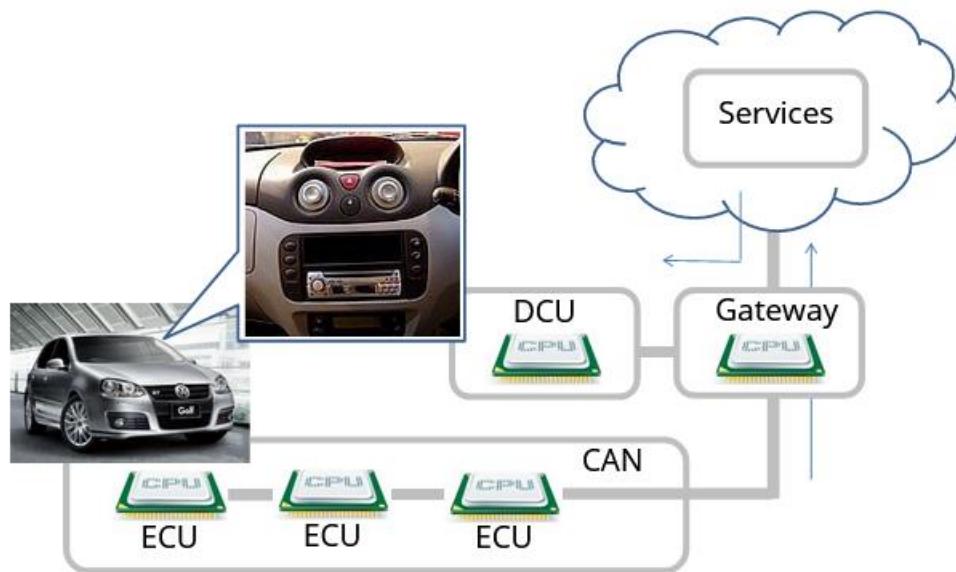
- Sigurnost i privatnost - veliki povezani sustav uvijek će biti ranjiviji od izoliranog
- Povezivanje svih fizičkih stvari na Internet
- Korištenje web protokola na baš svakom uređaju nekad nije najpametnije – recimo ograničenje kad se uređaj napaja samo baterijom – tad je možda bolje koristiti neki optimiziraniji IoT protokol

- [Web of Things \(WoT\) Architecture \(W3C Recommendation\)](https://www.w3.org/TR/2020/REC-wot-architecture-20200409/) - <https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>
- U kasnijim temamam proći ćemo i [Web of Things \(WoT\) Thing Description \(W3C Recommendation\)](#) te [Web of Things \(WoT\) Discovery \(W3C Working Draft\)](#)

- Namijenjen je omogućavanju interoperabilnosti preko IoT platformi i aplikacijskih domena.
- Sve u svemu, cilj WoT-a je očuvati i nadopuniti postojeće IoT standarde i rješenja.
- Općenito, W3C WoT arhitektura dizajnirana je da opiše što postoji, a ne da propisuje što implementirati.

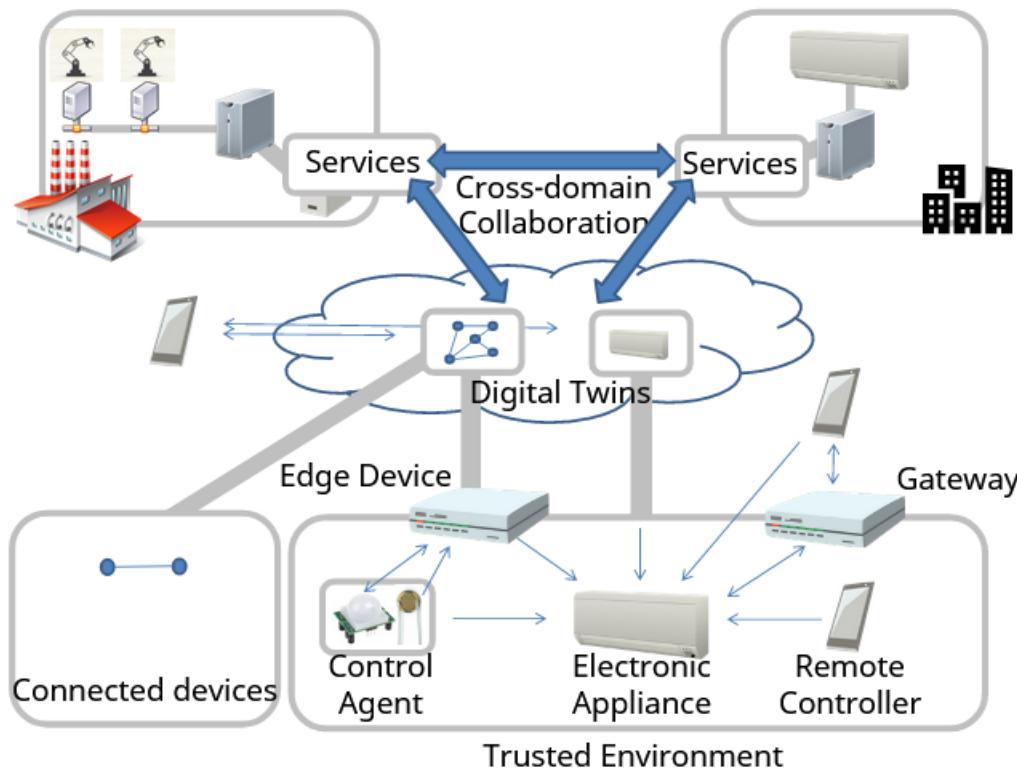
- opisuje apstraktnu arhitekturu web stvari.
- Ova apstraktna arhitektura temelji se na skupu zahtjeva koji su izvedeni iz slučajeva upotrebe za više domena aplikacija.
- Specifikacija opisuje kako su ti građevni blokovi povezani i rade zajedno.
- WoT apstraktna arhitektura definira osnovni konceptualni okvir koji se može preslikati na niz konkretnih scenarija implementacije, od kojih je dano nekoliko primjera.
- Međutim, apstraktna arhitektura opisana u specifikaciji sama po sebi ne definira konkretne mehanizme niti propisuje bilo kakvu konkretnu implementaciju.

# Primjer: Povezano vozilo



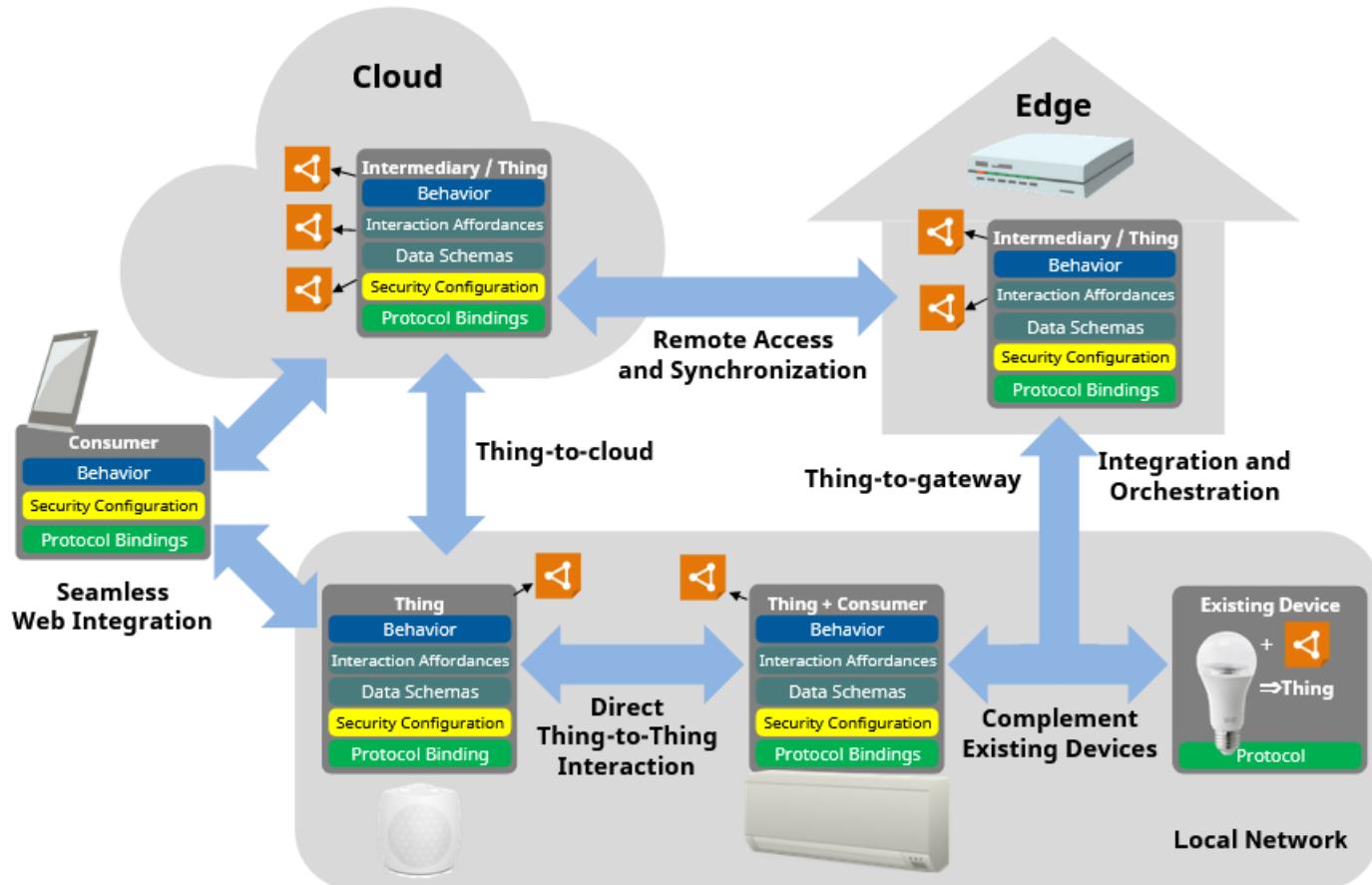
<https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>

# Pregled slučajeva korištenja iz W3C specifikacije WoT arhitekture



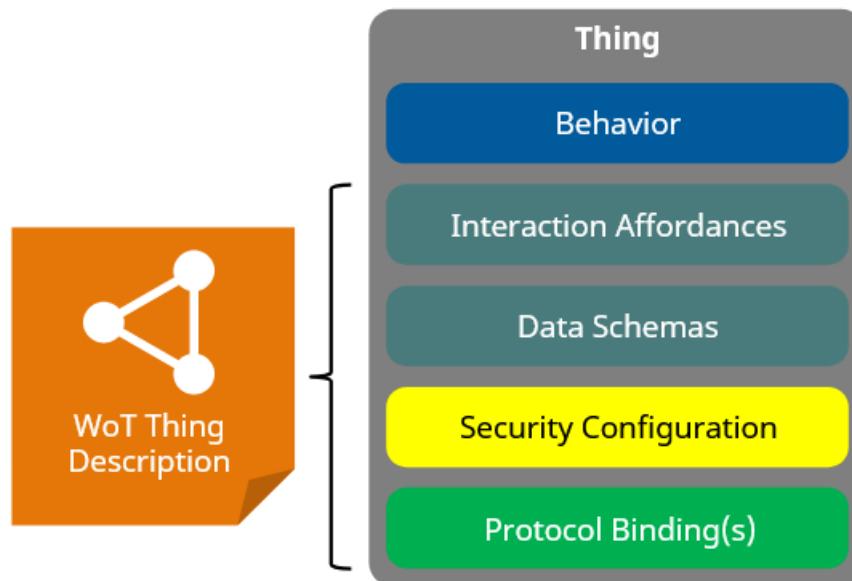
<https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>

# Abstraktna arhitektura W3C WoT



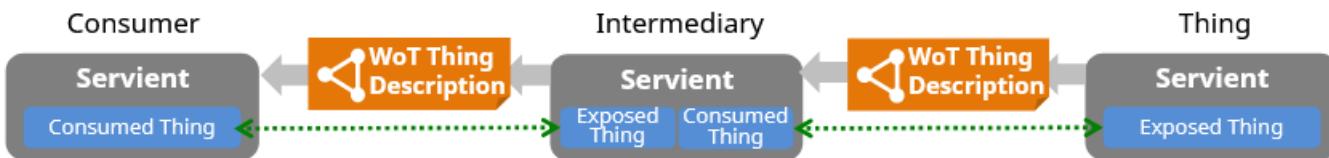
<https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>

# Arhitekturni aspekti pametne stvari



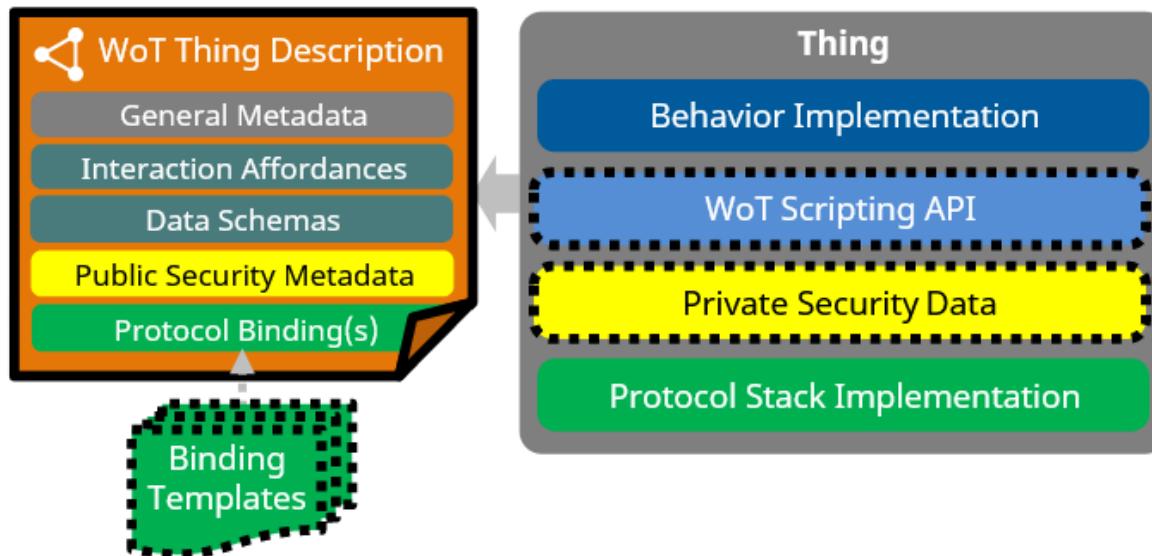
<https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>

# Direktna i indirektna komunikacija



<https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>

# Gradjevni elementi WoT-a



<https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>

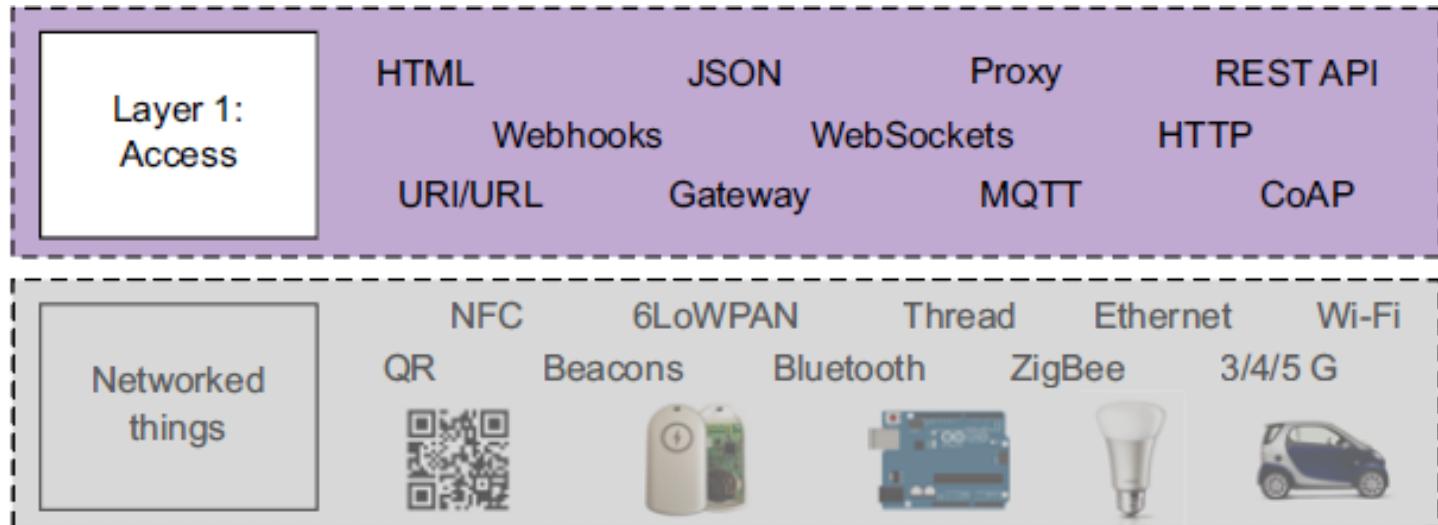


# Kreiranje Web API-ja za stvari

Darko Andročec

- Pojednostavljenje komunikacije među uređajima, servisima i aplikacijama korištenjem standarda sličnih ostalome na webu
- Implementacija web API-ja za fizičke objekte

# Povezivanje stvari na web



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

# Glavne korištene tehnologije

- REST ili RESTful

REST ograničenja:

- klijent-poslužitelj – zahtjev-odgovor predložak
- Jedinstveno sučelje
- Bez pamćenja stanja
- Mogućnost privremenog pohranjivanja (caching)
- Slojeviti sustav

- URL (Uniform Resource Locator) za resurs
  - <scheme> ":" <authority><path> [ "?" query ] [ "#" fragment ]
- U webu stvari <scheme> je uvijek http ili https
- <authority> je poslužitelj sa opcionalnih portom ili podacima za pristup
- <path> je bilo koji hijerarhijski put do resursa koji počinje sa /.
- Na kraju dolaze opcionalni parametri upita ili fragmenti
- Adresabilnost i prenosivost identifikatora resursa

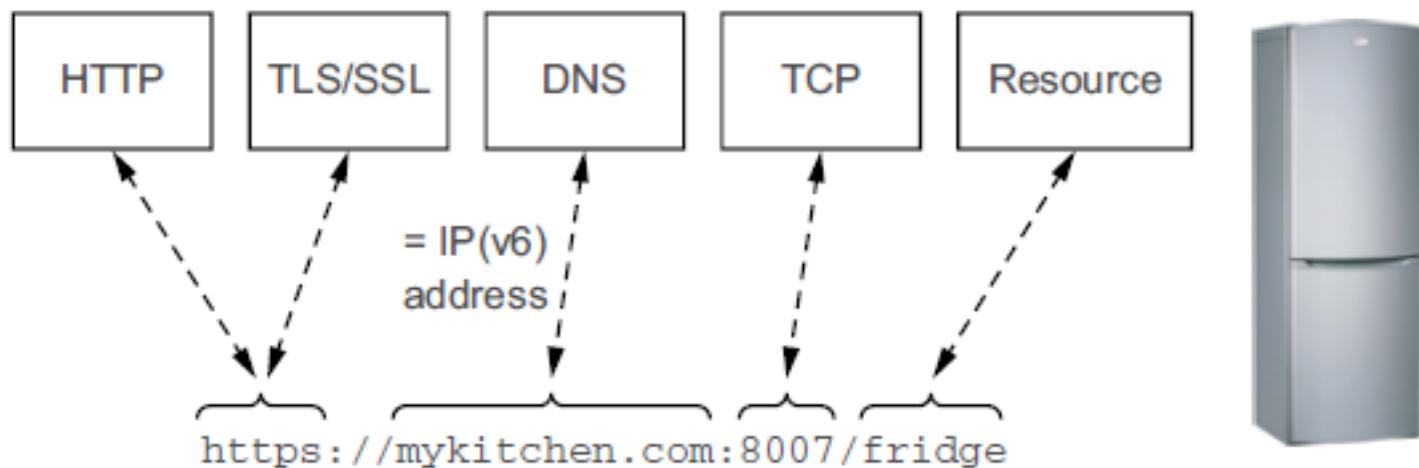
- Svaki uređaj weba stvari ima korijenski URL
- `http://gateway.api.com/devices/TV/`
- `http://kitchen-raspberry.device-lab.co.uk/`
- `https://192.168.10.10:9002/`
- `https://kitchen:3000/fridge/`

## Različiti tipovi resursa

- Mogu biti stvari (stvarni fizički resursi) i njihova stvarna svojstva, ali i kompletno virtualni
- # korisnik sa identifikatorom 12
- <https://webofthings.org/users/12>
- # uzorak br. 77654 iz listopada 2022
- <https://webofthings.org/samples/2022/10/77654>
- # Uredjaj s imenom svjetiljka12
- <https://devices.webofthings.io/svjetiljka12>

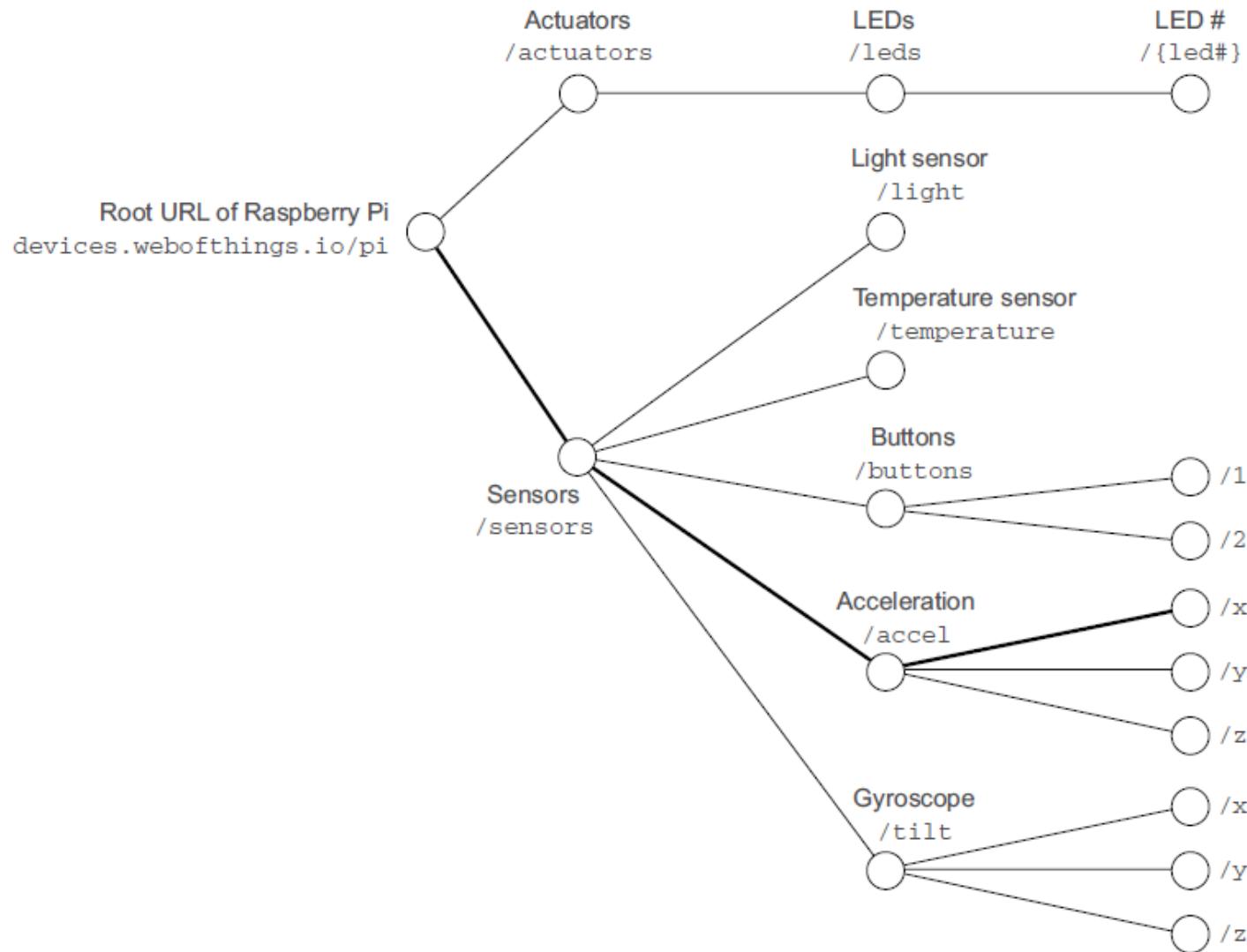
- Resursi su na webu često organizirani u hijerarhije definirane putem (engl. *path*)
- # lista senzora na uređaju (svi senzori na uređaju s ID-jem 24)
- <http://devices.webofthings.io/24/sensors>
- # lista uređaja na nekom području (zgrada 4)
- <http://192.168.44.12/zgrada4/devices/>
- # lista očitanja sa senzora
- <https://webofthings.org/devices/4554/samples>

# Čitanje URL-a stvari (1)



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

# Čitanje URL-a stvari (2)



- Koristi opisna imena
- Ne koristi glagole u URL-ovima – glagoli se koriste za HTTP metode, a ne za URL-ove
- Koristi množinu za agregirajuće resurse – npr. ako stvar ima nekoliko senzora, oni mogu biti dostupni putem resursa roditelja nazvanog /sensors

- Web stvar treba biti HTTP poslužitelj – ako se uređaju ne može poslati HTTP zahtjev, onda on nije dio weba stvari
- Web stvari trebaju koristiti sigurnu HTTP vezu (HTTPS)
- Web stvar treba imati korijenski resurs dostupan preko HTTP URL-a
- Web stvari moraju izložiti svoja svojstva pomoću hijerarhijske strukture

# Primjeri HTTP zahtjeva i odgovora (1)

**Request:**

```
GET /pi  
Host: devices.webofthings.io  
Accept: text/html
```

**Request headers****Response:**

```
200 OK HTTP/1.1  
Content-Type: text/html
```

**Response headers**

```
<html>  
...
```

**Response body**

Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

# Primjeri HTTP zahtjeva i odgovora (2)

**Request:**

```
GET /pi
Host: devices.webofthings.io
Accept: application/xml
```

**Response:**

```
200 OK
Content-Type: application/xml

<device>
  <name>Pi</name>
  ...
</device>
```

**Request:**

```
GET /pi
Host: devices.webofthings.io
Accept: application/json
```

**Response:**

```
200 OK
Content-Type: application/json

{
  "name" : "Pi"
  ...
}
```

- Web stvari trebaju podržavati JSON kao podrazumijevanu reprezentaciju
- Web stvari podržavaju UTF8 enkodiranje za zahtjeve i odgovore
- Web stvari opcionalno mogu ponuditi i HTML sučelje/reprezentaciju (UI)

- Princip samoopisujućih poruka
- Fiksni skup operacija koje svaki resurs može podržati – najčešće GET, POST, PUT, DELETE, HEAD
- GET – operacija samo za čitanje
- POST- kreiranje nove instance nečega što još nema svoj URL
- PUT – metoda za ažuriranje
- DELETE – brisanje resursa

## Primjer: GET

- Čitanje resursa (temperaturni senzor na Raspberry Pi)

### Request:

```
GET /pi/sensors/temperature/value  
Accept: application/json  
Host: devices.webofthings.io
```

### Response:

```
200 OK HTTP/1.1  
Content-Type: application/json  
  
{ "temperature" : 37 }
```

## Primjer: POST

- Kreiranje novog resursa

Request:

```
POST /pi/display/messages HTTP/1.1
Host: devices.webofthings.io
Content-Type: application/json

{ "content": "Hello World!", "duration": 30 }
```

Response:

```
201 Created HTTP/1.1
Location: devices.webofthings.io/pi/display/messages/2210
```

A POST request contains a payload, here of MIME type application/json.

The server replies with a 201 status code meaning that the rule was created instantly.

The response header contains the URL of the newly created rule.

Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

## Primjer: PUT

- Ažuriranje postojećeg resursa – promjena boje LED-ice

### Request:

```
PUT /pi/actuators/leds/4 HTTP/1.1
```

```
Host: devices.webofthings.io
```

```
Content-Type: application/json
```

```
{"red": 0, "green": 128, "blue": 128}
```

### Response:

```
200 OK HTTP/1.1
```

Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

## Primjer: DELETE

- Brisanje postojećeg resursa

**Request:**

```
DELETE /rules/24 HTTP/1.1  
Host: devices.webofthings.io
```

**Response:**

```
200 OK HTTP/1.1
```

- Web stvari trebaju podržavati GET, POST, PUT i DELETE HTTP metode
- Web stvari trebaju podržati HTTP kodove o statusu 20x, 40x i 50x
- Web stvari moraju podržavati GET metodu na njihovom korijenskom URL-u
- Web stvari trebaju podržavati CORS – CORS (cross-origin resource sharing) služi za omogućavanje Javascripta na strani klijenta za pristup resursima

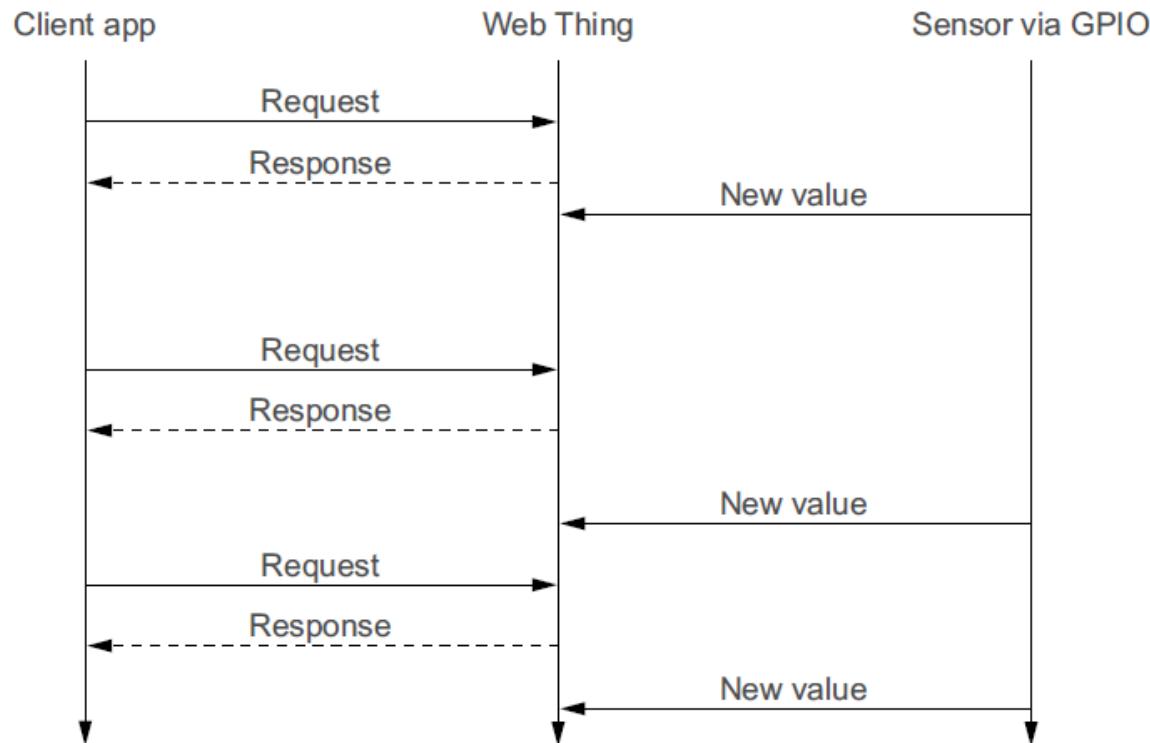
## Povezivanje resursa

- Web stvari bi trebale podržavati mogućnost pretraživanja sa linkovima - trebaju uvijek ponuditi poveznice na resurse koji se s njima u hijerarhiji resursa, posebno na resurse roditelja i djece u stablu hijerarhije
- Web stvari mogu podržavati OPTIONS metodu za svaki od njihovih resursa

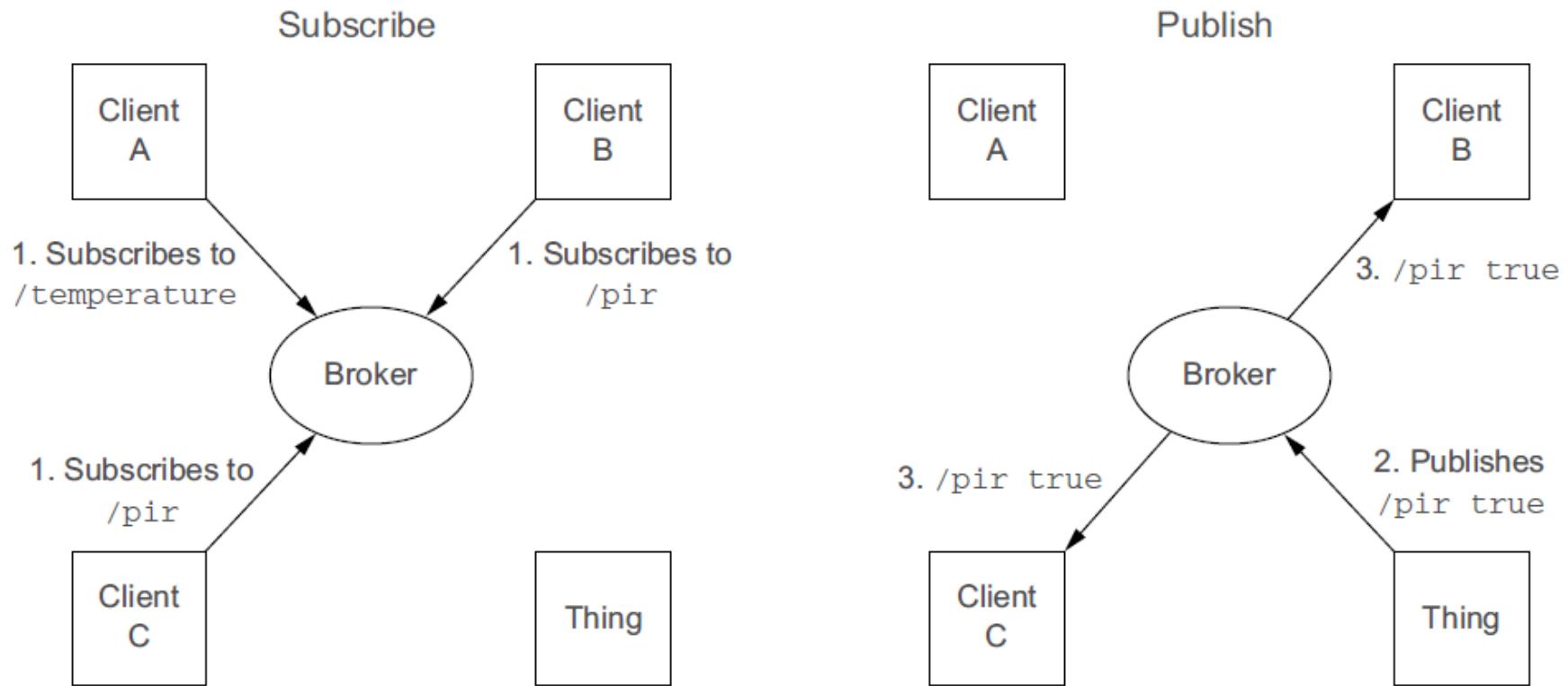
- Model koji pokreće klijent nije praktičan za aplikacije u kojima uređaj mora asinkrono slati obavijesti klijentima čim se proizvedu.
- Na primjer, sigurnosna kamera ili požarni alarm moraju biti u mogućnosti odmah poslati upozorenje kada se otkrije bilo kakva anomalija i ne bi trebali čekati dok klijent ne zatraži te informacije.

# Osnovno glasanje (basic polling)

- Klijentska aplikacija šalje zahtjeve prema web stvari u pravilnim intervalima. Rezultati koje klijentska aplikacija dobiva nisu sinkronizirani sa novim vrijednostima senzora.



# Objavi/preplati se (Publish/subscribe)



- Webhooks - HTTP povratni pozivi (HTTP callbacks)
- Comet
- WebSockets

- Najjednostavniji način
- Svaki entitet je ujedno i klijent i poslužitelj
- Na ovaj način i web stvari i web aplikacije mogu djelovati kao HTTP klijenti inicirajući zahtjeve drugim poslužiteljima, a mogu imati i poslužitelj koji može odgovoriti na druge zahtjeve u isto vrijeme.
- Sve što trebamo je implementirati REST API i na stvar i na klijenta, koji tada također postaje poslužitelj.
- Ograničenje: pretplatnik mora imati javno dostupnu URL ili IP adresu.

# Webhooks – primjer (1)

Client Request:

```
POST /pi/sensors/humidity/subs HTTP/1.1  
Host: devices.webofthings.io  
Content-Type: application/json
```

```
{"callback" : "https://url-of-client-a.com/pubs"}
```

The client subscribes via a POST request on the Thing to the humidity events.

Thing Response:

```
201 Created HTTP/1.1  
Content-Type: application/json  
Location: devices.webofthings.io/pi/sensors/humidity/subs/1234
```

```
{"humidity": 37}
```

The client provides a callback URL that will be used by the Thing to push the event. Note that sometimes the Referer header is used to specify the callback, but because the header was intended for a very different purpose, we recommend not using it in another way than intended.

As a good practice, the Thing also sends the current humidity value to spare another request.

The Thing responds with a reference to the newly created subscription.

# Webhooks – primjer (2)

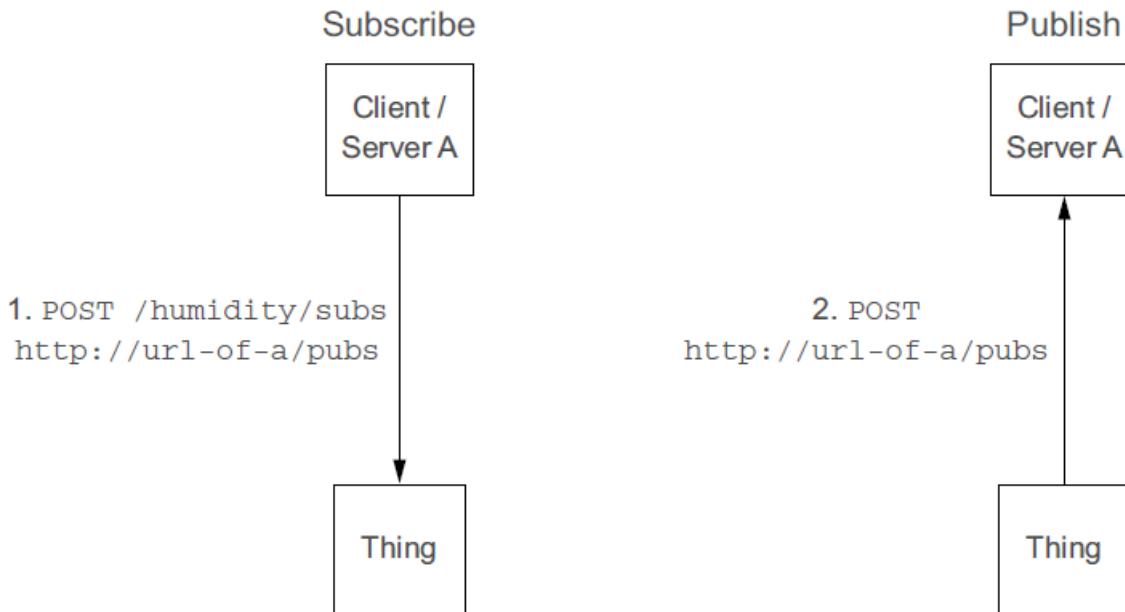
The referrer header contains the URL of the subscription.

## Thing Request:

```
POST /pubs HTTP/1.1  
Host: url-of-client-a.com  
Referer: http://devices.webofthings.io/pi/sensors/humidity/subs/1234  
Content-Type: application/json  
  
{ "humidity" : 50 }
```

When a new humidity value is available, the Thing creates a POST request on the client API.

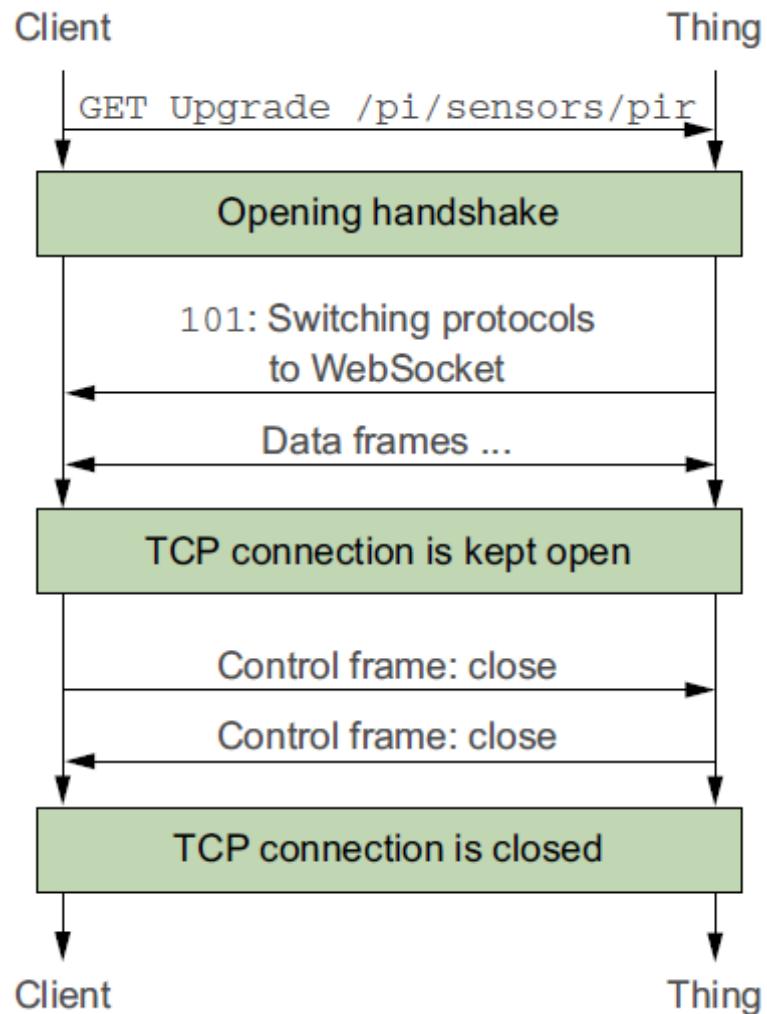
The payload contains the new humidity value, pushed to the client.



- Niz tehnika za zaobilaženje ograničenja HTTP pollinga i webhooksa predstavljanjem komunikacije osnovane na događajima preko HTTP-a
- Jedna od tehnika je dugo glasanje (*long polling*):
- Klijent šalje standardni HTTP zahtjev poslužitelju, ali umjesto da odmah primi odgovor, poslužitelj zadržava zahtjev dok se od senzora ne primi događaj, koji se zatim ubacuje u odgovor vraćen na klijentov zahtjev koji je bio u stanju mirovanja.

- Comet tehnike su zakrpa a ne rješenje -> neučinkoviti su
- WebSocket je dio HTML5 specifikacije
- Sve veća podrška za HTML5 u najnovijim web i mobilnim web preglednicima znači da WebSocket postaje sveprisutno dostupan svim web aplikacijama.
- Dobar je kandidat za pub/sub implementaciju u webu stvari
- WebSockets omogućuje full-duplex komunikacijski kanal preko jedne TCP veze.
- Nakon što se napravi početno rukovanje, klijent i poslužitelj moći će slati poruke natrag i naprijed preko otvorene TCP veze; ove poruke nisu HTTP poruke već podatkovni okviri WebSocketsa.

# WebSockets rukovanje



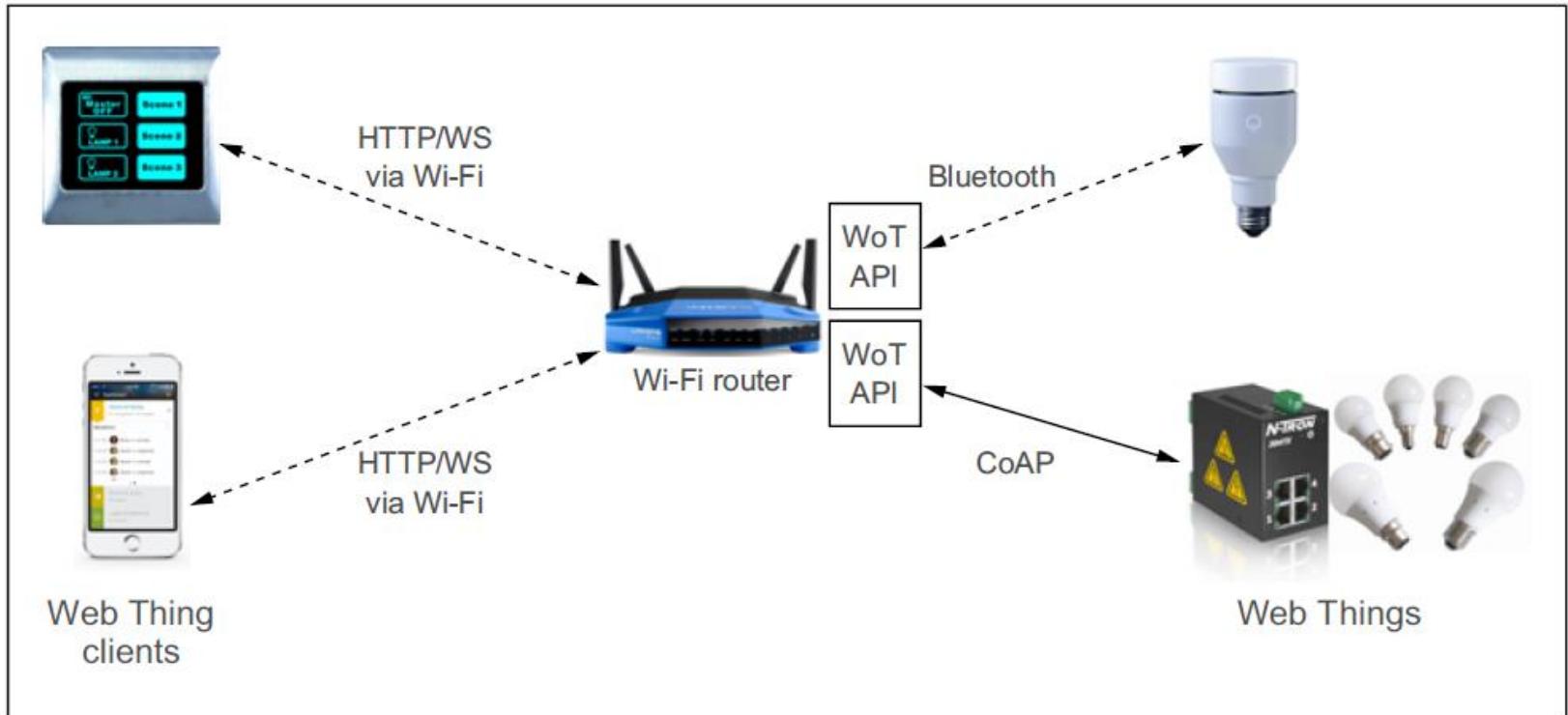
- WebSockets idu preko porta 80 koji je najčešće otvoren (nije blokiran od strane vatrozidova ili proxy-ja)
- Hijerarhijska struktura web stvari može se ponovno iskoristiti i u WebSocketima
- WebSockets se može implementirati u bilo kojoj platformi koja podržava TCP/IP
- Nedostatak stalnog držanja otvorene TCP veze – povećanje korištenja baterije i teže skaliranje na poslužiteljskoj strani nego što je to slučaj kod HTTP-a

- Direktna integracija
- Integracija pomoću pristupnika (gatewaya)
- Integracija pomoću oblaka

- REST na uređaju
- Ovo zahtijeva da je stvar dostupna putem Internet protokola i da može biti HTTP poslužitelj
- Najbolji je izbor kad stvar nije pogonjena baterijom i kad se zahtijeva direktni pristup klijentima (npr. odgovarajuća mobilna aplikacija) – npr. automatizacija kuće (pametna kuća) – WiFi povezane pametne svjetiljke

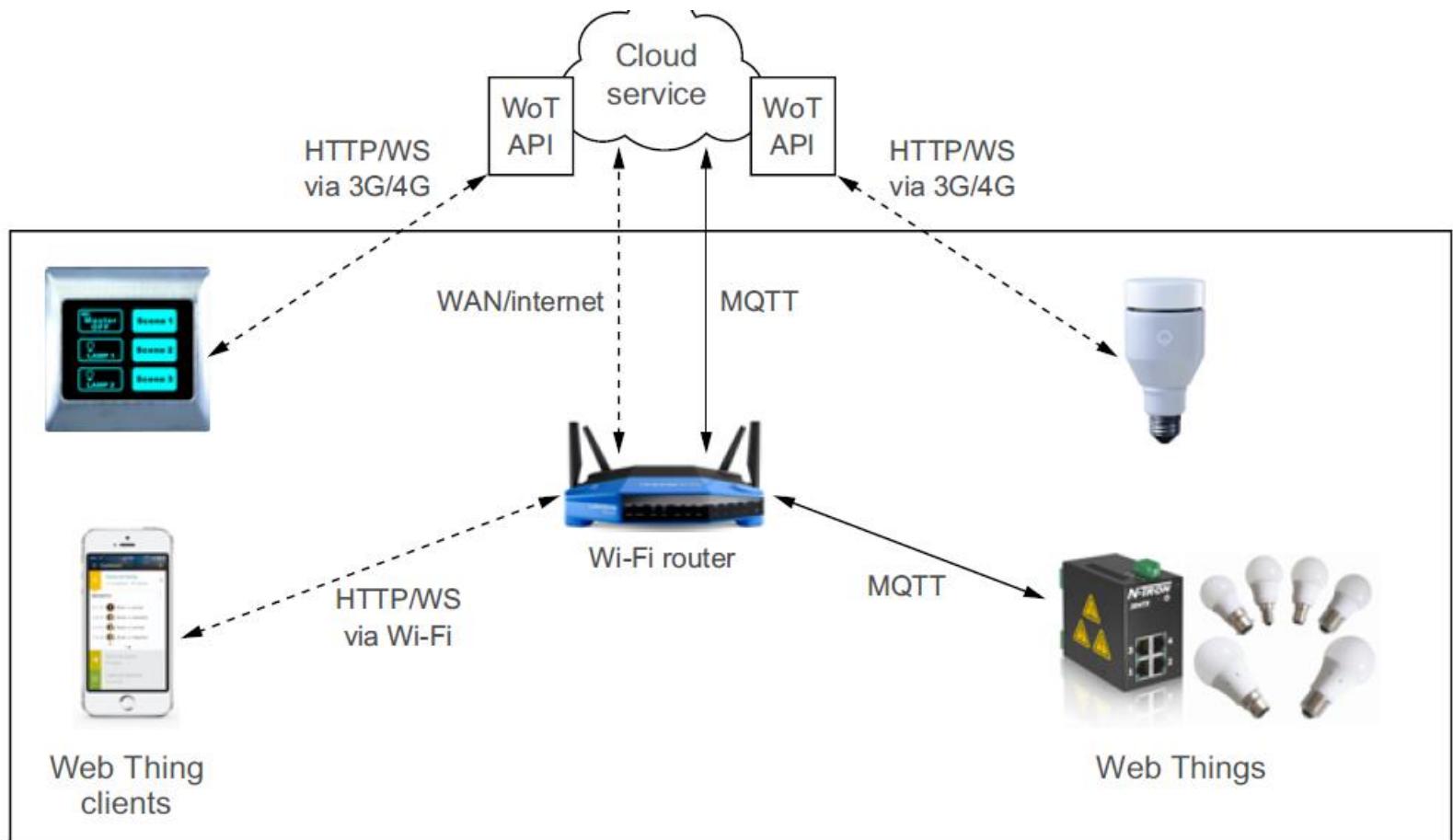
- Nemaju svi IoT uređaji ugrađenu podršku za HTTP ili TCP/IP
- Uređaji koji se napajaju baterijama često ne podržavaju Wi-Fi ili Ethernet, već podržavaju ZigBee ili Bluetooth
- Takvi uređaji također mogu biti dio weba stvari sve dok negdje postoji posrednik koji može otkriti funkcionalnost uređaja putem WoT API-ja poput onog koji smo prethodno opisali.
- Ti se posrednici nazivaju aplikacijski pristupnici (application gateways)

# Integracija pomoću pristupnika (2)



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

# Integracija pomoću oblaka (1)



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

## Integracija pomoću oblaka (2)

- Stvar ne može izravno ponuditi web API.
- Ali usluga u oblaku djeluje kao moći aplikacijski pristupnik, nudeći mnogo više značajki u ime stvari.
- U ovom konkretnom primjeru, web stvar povezuje se putem MQTT-a s uslugom u oblaku, koja izlaže API web stvari putem HTTP-a i WebSockets API-ja.
- Usluge u oblaku također mogu ponuditi mnoge dodatne značajke kao što su neograničena pohrana podataka, upravljanje korisnicima, vizualizacija podataka, obrada toka, podrška za mnoge istodobne zahtjeve i još mnogo toga.

# Pitanja i rasprava



# Protokoli IoT-a

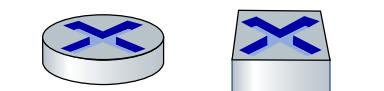
Nikola Ivković

# Prisjetimo se osnovnih elemenata Interneta



Milijarde povezanih računalnih *uređaja*:

- *domaćini* = krajnji sustavi
- izvode mrežne aplikacije



*Prespajanje paketa*  
(komadi podataka)

- usmjernici, prespojnici



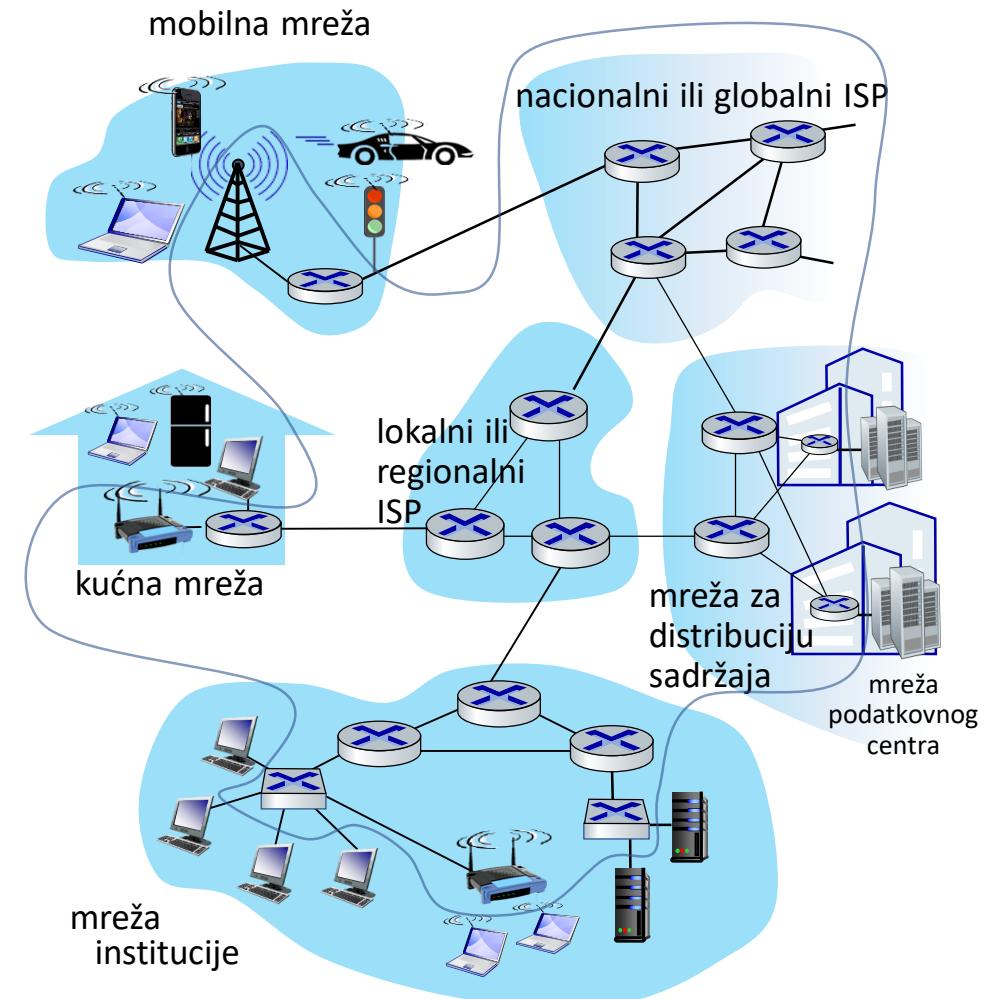
*Komunikacijske poveznice*

- optika, bakar, radiovalovi, sateliti
- brzina prijenosa: *bandwidth*



*Mreže*

- Grupa uređaja, usmjernika i poveznica – njima upravlja organizacija



# Mrežne aplikacije odn. distribuirane aplikacije

- Dva računalna procesa koja se izvode na različitim računalima domaćinima i komuniciraju s pomoću računalne mreže (npr. Interneta)
- Klasične mrežne aplikacije
  - e-pošta, Web, prijenos datoteka, ...
- Koje su specifičnosti IoT-a i kakvi protokoli nam trebaju?
  - Jednostavni uređaji, ponekad vrlo ograničenih mogućnosti
  - Bežična komunikacija je manje pouzdana od vodova
  - ...

# Zabavni uređaji povezani na Internet



Amazon Echo



Hladnjak povezan na Internet



Sigurnosna kamera



Internetski telefoni



IP okvir za slike



Slingbox: remote control cable TV



Uređaji za igre



Pacemaker & Monitor



Web-podržani tosteri+ vremenska prognoza



madrac sa senzorima



Fitbit



Tweet-a-watt:  
Nadgledanje potrošnje energije

bicikli



autobobili



romobili

*Drugi?*

# Moderni Internetov protokolni slog („TCP/IP model”)

- **aplikacijski (application):** podrška mrežnim aplikacijama
  - HTTP, IMAP, SMTP, DNS, FTP
- **transportni (transport):** prijenos podataka *od procesa do procesa*
  - TCP, UDP
- **mrežni (network):** usmjeravanje datagrama od izvora do odredišta
  - IP, protokoli usmjeravanja
- **poveznički (link):** prijenos podataka između dva susjedna mrežna elementa
  - Ethernet, 802.11 (WiFi), PPP
- **fizički (physical):** bitovi “na žici”



# ISO/OSI referentni model

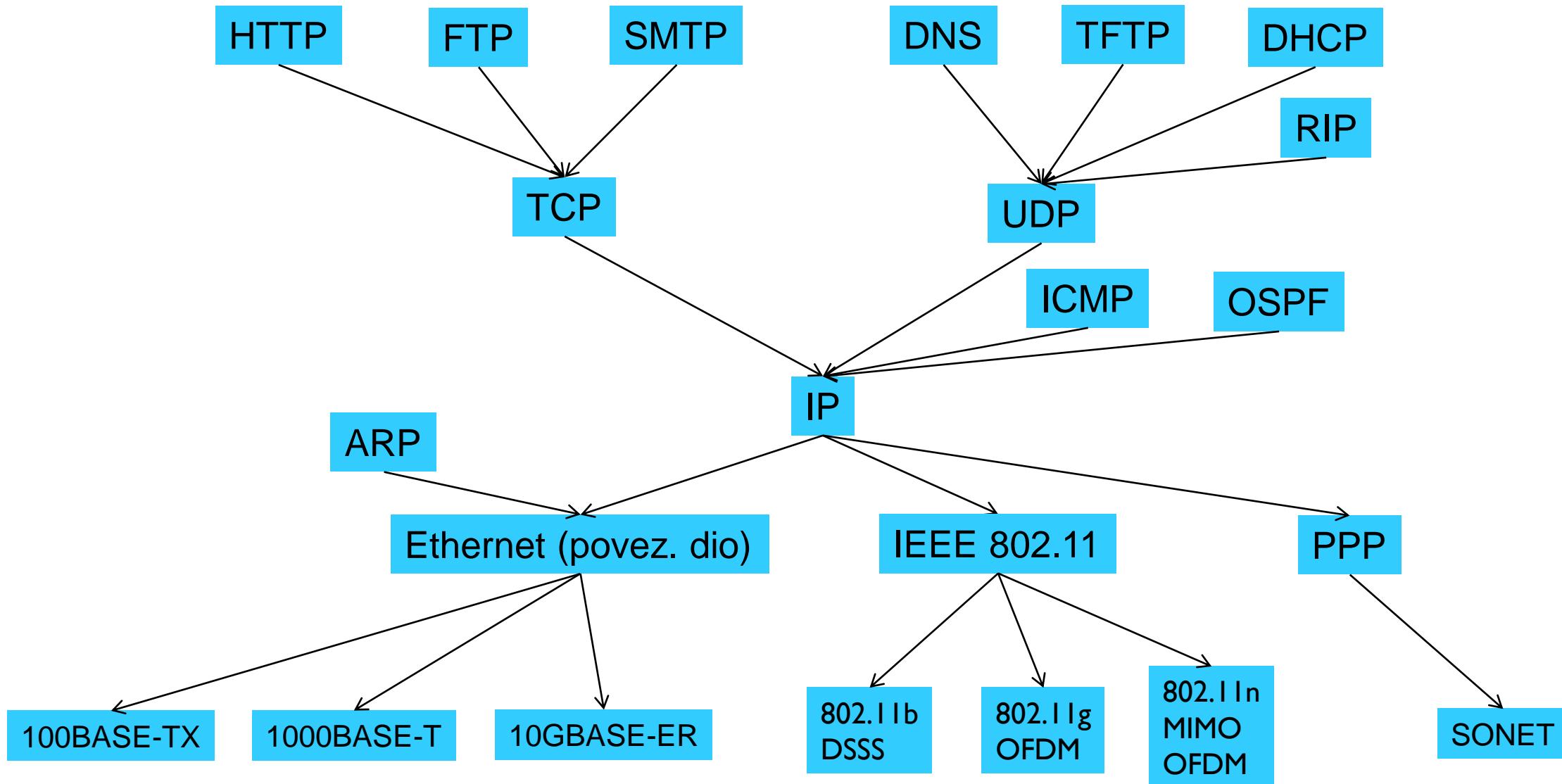
Sadrži dva sloja kojih nema u modernom Internetskom modelu!

- *prikaz (presentation)*: omogućava aplikacijama da interpretiraju značenje podataka, npr., enkripcija, kompresija, kodiranja
- *razgovora (session)*: sinkronizacija, provjera stanja, oporavak od pogrešaka u razmjeni podataka
- Internetovom modelu “nedostaju” ovi slojevi!
  - ove se usluge, ako su *potrebne*, moraju implementirati npr. u aplikacijskom ili transportnom sloju



Sedmoslojni OS/ISO  
referentni model

# Primjeri učahurivanja protokola - oblik pješčanoga sata



# Na kojim bi slojevima bilo dobro imati specijalizirane protokole za IoT?

- *aplikacijski*: podrška mrežnim aplikacijama

- HTTP
- MQTT, CoAP, AMQP, DDS, LwM2M

- *poveznički i fizički*: prijenos podataka između dva susjedna mrežna elementa

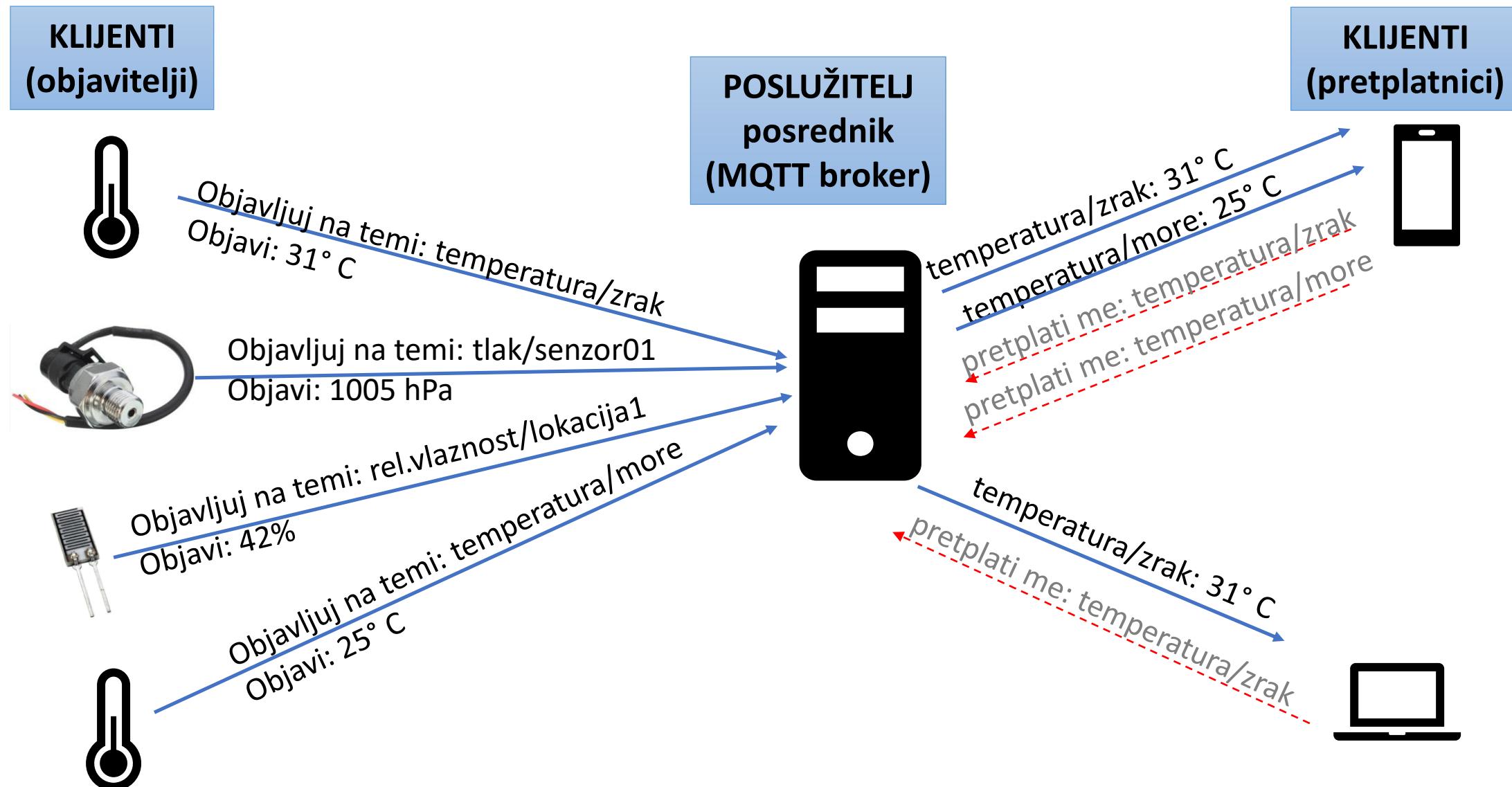
- Wi-Fi (IEEE 802.11), Bluetooth
- ZigBee, LoRa, LoRaWAN, LTE CAT 1, LTE CAT M1, NB-IoT
- RFID



# MQTT - *Message Queueing Telemetry Transport*

- Aplikacijski protokol
- Arhitektura klijent/poslužitelj
- Poslužitelj je posrednik – MQTT broker
  - posreduje između različitih klijenata
- Dvije uloge klijenata
  - objavitelj (eng. *publisher*)- objavljuje poruke koje se preko poslužitelja dojavljaju drugima
  - preplatnik (eng. *subscriber*) – prijavljuje se za primanje nekih tipova poruka

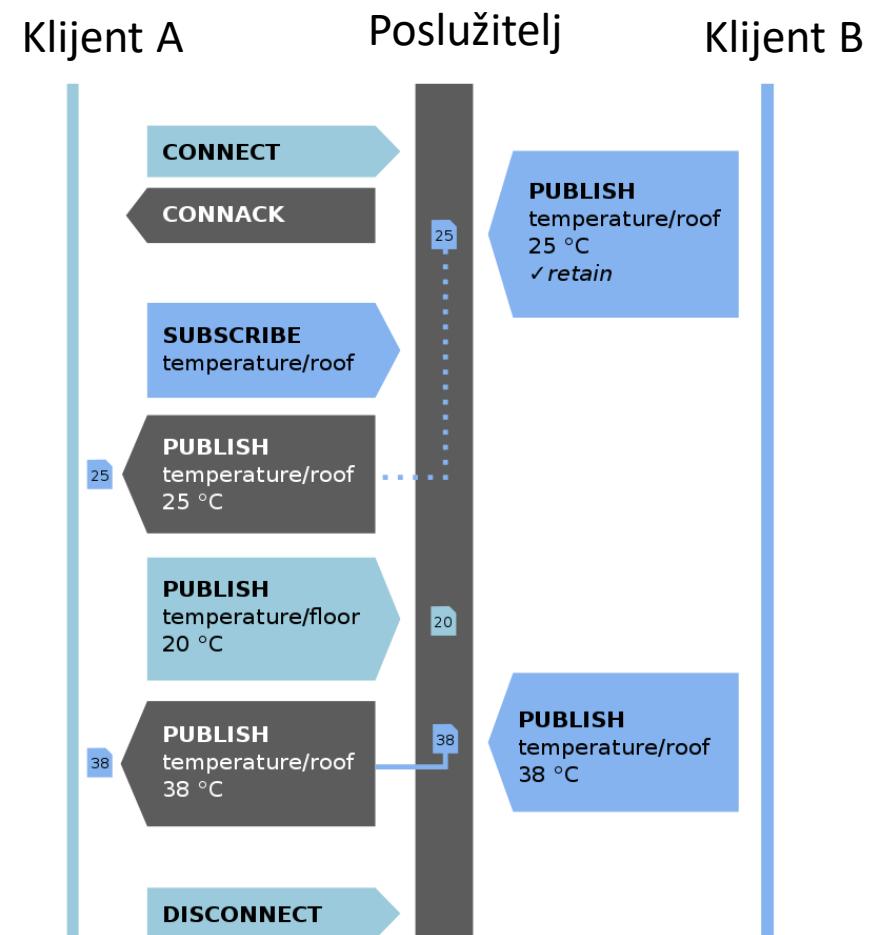
# MQTT – primjer arhitekture objavi/pretplati se



# Razmjena poruka – primjer

- Klijent A uspostavlja vezu s poslužiteljem
- Klijent B objavljuje temperaturu krova ( $25^{\circ}\text{C}$ )
- Klijent A se preplaćuje na temperaturu krova
  - dobiva poruku koja je stigla do brokera prije nego se preplatio jer se koristi zastavicu RETAIN
- Klijent B objavljuje, a klijent A dobiva temperaturu krova  $38^{\circ}\text{C}$
- Klijet A raskida vezu s poslužiteljem

Slika preuzeta s [https://en.wikipedia.org/wiki/MQTT#/media/File:MQTT\\_protocol\\_example\\_without\\_QoS.svg](https://en.wikipedia.org/wiki/MQTT#/media/File:MQTT_protocol_example_without_QoS.svg)



# Format MQTT-ovih poruka

- Zaglavje: fiksni dio(2 bajta) i varijabilni dio (opcionalno)
- Iza zaglavlja teret (eng. *payload*) – nemaju svi tipovi poruka
- 1. bajt fiksnog dijela
  - $2^4 = 16$  tipova poruka, od toga se koristi 15
  - Zastavice ovisno o tipu poruke: PUBLISH koristi DUP (3), QoS (2-1), RETAIN (0)
- 2. bajt duljina ostatka poruke u bajtovima (varijabilni dio zaglavlja + teret)
- U varijabilnom dijelu
  - mnoge poruke imaju identifikator paketa od 2 bajta

Bit	7	6	5	4	3	2	1	0
byte 1	MQTT Control Packet type				Flags specific to each MQTT Control Packet type			
byte 2...	Remaining Length							

# Popis tipova poruka: 0 - 7

Name	Value	Direction of flow	Description
Reserved	0	Forbidden	Reserved
CONNECT	1	Client to Server	Connection request
CONNACK	2	Server to Client	Connect acknowledgment
PUBLISH	3	Client to Server or Server to Client	Publish message
PUBACK	4	Client to Server or Server to Client	Publish acknowledgment (QoS 1)
PUBREC	5	Client to Server or Server to Client	Publish received (QoS 2 delivery part 1)
PUBREL	6	Client to Server or Server to Client	Publish release (QoS 2 delivery part 2)
PUBCOMP	7	Client to Server or Server to Client	Publish complete (QoS 2 delivery part 3)

# Popis tipova poruka: 8 - 15

Name	Value	Direction of flow	Description
SUBSCRIBE	8	Client to Server	Subscribe request
SUBACK	9	Server to Client	Subscribe acknowledgment
UNSUBSCRIBE	10	Client to Server	Unsubscribe request
UNSUBACK	11	Server to Client	Unsubscribe acknowledgment
PINGREQ	12	Client to Server	PING request
PINGRESP	13	Server to Client	PING response
DISCONNECT	14	Client to Server or Server to Client	Disconnect notification
AUTH	15	Client to Server or Server to Client	Authentication exchange

# Transport MQTT poruka

- MQTT preko TCP-a
- MQTT preko TLS-a (i TCP-a)
- MQTT-SN (*MQTT for Sensor Networks*) je varijanta za senzorske mreže koje koriste baterijsko napajanje.
  - transport poruka preko UDP-a ili Bluetootha
  - prisjetimo se: Bluetooth nije internetski tip mreže

# Još neki detalji o MQTT-u

## QoS

- Najviše jednom odn. „pošalji i zaboravi”
- Najmanje jednom ond. potvrda dostave – ponavlja se slanje dok se ne dobije potvrda
- Točno jednom odn. osigurana dostava – pošiljatelj i primatelj koriste „dvorazinsko rukovanje”

## Kako implementirati sustav s pomoću MQTT-a?

- Postoji mnoštvo biblioteka za MQTT za razne programske jezike i platforme
- Kao i kod ostalih protokola – može se samostalno implementirati formiranjem poruke i slanjem preko TCP socketa

# Drugi protokoli aplikacijskog sloja (1/3)

## HTTP

- Napravljen za Web
- Prolazi kroz praktički svaki vatroštit (eng. *firewall*)
- Protokol bez stanja, ASCII zaglavje varijabilne duljine
- REST API
- Nije pogodan za jednostavne uređaje koji koriste bateriju

## CoAP (Constrained Application Protocol)

- Namijenjen za uređaje sa slabim resursima npr. uređaji za bežične senzorske mreže (WSN)
- RFC 7252
- Transportni protokol UDP ili DTLS (i UDP) => CoAP implementira mehanizme pouzdanog prijenosa, kontrole toka i kontrole zakrčenja
- RFC 8323: CoAP preko TCP-a, TLS-a i WebSocketsa

# Drugi protokoli aplikacijskog sloja (2/3)

## AMQP (*Advanced Message Queuing Protocol*)

- Nije dizajniran specifično za IoT
- Važna interoperabilnost i fleksibilnost
- Asinkrona komunikacija
- Veća kompleksnost nego npr. MQTT, a zahtjeva i više resursa

## DDS (*Data Distribution Service*)

- Razvijeno za pouzdane, sigurnosno-kritične sustave koji zahtijevaju rad u realnom vremenu
- Visoke performanse i interoperabilnost
- Uzorak objave/preplate
- Može koristiti transportne protokole TCP i UDP

# Drugi protokoli aplikacijskog sloja (3/3)

## XMPP (*Extensive Messaging and Presence Protocol*)

- Klijent/poslužitelj arhitektura sa identifikatorima sličnim kao e-mail adrese
- Zasnovano na XML-u
- Izvorno dizajniran za tekstualne instant poruke (IM)
- Nije pogodno za uređaje koji imaju vrlo ograničene resurse

## LwM2M (*Lightweight Machine-to-Machine*)

- Nastao nadogradnjom CoAP
- Klijent/poslužitelj arhitektura
- Udaljeno upravljanje uređajem npr. nadogradnja firmwarea
- Za transport poruka može koristiti UDP, TCP ili čak SMS

# Protokoli povezničkog i fizičkog sloja

- Bežične poveznice
- Različiti dometi, brzine prijenosa, potrošnja energije, različite tehnike
- Infrastrukturno vs ad hoc
- Jedan skok vs više skokova
- Dijeljenje zajedničkog komunikacijskog medija
  - protokoli s podjelom kanala LTE, GSM, NB-IoT
  - protokoli sa slučajnim pristupom: Wifi, LoRa, ZigBee, BLE
  - protokoli tipa naizmjence: Bluetooth

# Velike udaljenosti: mobilne komunikacije

## Mobilne komunikacije 2G, 3G, 4G (LTE), 5G

- Velike udaljenosti (npr. 10-20km)
- Velika brzina prijenosa
- SIM kartica
- Infrastrukturu nude telekomunikacije tvrtke
- Nije pogodno za uređaje koji se napajaju baterijom
- Razmjerno skupa usluga

## LTE CAT 1, LTE M

- Varijante LTE-a koje su prilagođene IoT-a
- Jednostavnije i jeftinije od običnog LTE-a
- I dalje troši dosta energije pa nije pogodno za uređaje koji se napajaju iz baterije

## NB-IoT (Narrowband IoT)

- LTE s manjom potrošnjom energije, manjom brzinom, manjom cijenom

# Velike udaljenosti: LoRa i LoRaWAN

## LoRa (Long Range)

- Samo fizički sloj
- Širenje spektra (spread spectrum)
- Ispravljanje pogrešaka (*Error correction code*)
- Frekvencijski pojasevi
  - Europa: 863–870/873 MHz
  - Južna Amerika: 915–928 MHz
  - Sjeverna Amerika: 902–928 MHz
  - Indija: 865–867 MHz
  - Azija 915–928 MHz
  - svijet 2.4 GHz
- 0.3 kbit/s and 27 kbit/s
- Mogu koristiti uređaji s baterijom

## LoRaWAN

- Koristi LoRa
- Poveznički sloj
- Ažuriranje firmwarea „over-the-air”
- Gateway
- Klasa A - osnovna/predefinirana klasa za najmanju potrošnju energije
  - uređaj može spavati bez potrebe za periodičkim buđenjem
  - dobro za uređaje s baterijom
- Klasa B - uređaju se periodički sinkroniziraju s *beacon* okvirom
- Klasa C – najmanja latencija jer je primatelj uvijek uključen
  - troši više energije pa je prikladno za uređaje koji koriste napajanje iz električne mreže

# Lokalne bežične mreže – IEEE 802.11 (WiFi)

- IEEE 802.11 (WiFi)
  - tipičan domet u desecima metara (s uobičajenim antenama i snagama signala)
  - CSMA i CSMA/CA (RTS i CTS)
  - pouzdan prijenos – okviri potvrde (eng. ACK)
- Različiti fizički standardi (različite brzine i načini kodiranja)
  - npr. IEEE 802.11b, IEEE 802.11g, IEEE 802.11n, IEEE 802.11ac, IEEE 802.11ax,...
- Poveznički sloj – svi imaju isti format okvira
- Infrastrukturni i *ad hoc* način rada
- Infrastrukturno
  - bazna stanica zvana i pristupna točka (eng. *access point*)
  - BSSID
- Kontrola snage – štedi energiju, bežični čvorovi mogu spavati između periodičnih *beacon* okvira koje odašilje pristupna točka

# Manji domet: Bluetooth & ZigBee

## Bluetooth

- Na prati Internetski model
- Cijeli protokolni stog
- Aplikacijski profili
- IEEE standardizirao fizički i poveznički sloj
- MAC protokol kod kojeg se kanal dijeli naizmjence
  - *prozivanje (eng. Polling)* po principu *master-slaves*)
- BLE (*Bluetooth Low Energy*)
  - mala potrošnja energije
  - nema kompatibilnosti s Bluetoothom, ali se dizajnirani da si međusobno što manje smetaju
  - tehnika širenja spektra

## ZigBee

- Dizajnirana da troši vrlo malo energije (manje od Bluetootha)
- IEEE 802.15.4
- Primjena
  - bežične senzorske mreže
  - povezivanje jednostavnih uređaja u kući (senzori, prekidači, rasvjeta...)
  - ...
- Tri klase uređaja
  - koordinator
  - usmjernik
  - krajnju uređaj
- CSMA/CA

# RFID

- Automatska identifikacija objekata je korisna za IoT
- Objekti imaju oznaku (eng. *tag*)
- Mogu biti i pasivni (bez svoz izvora napajanja)
  - pobudi ih čitač svojim elektromagnetskim valovima i da im dovoljno energije za komunikaciju
  - jednostavna funkcija – često samo javlja svoju identifikacijsku oznaku

# Za kraj

- Protokoli uglavnom na aplikacijskom te fizičkom i povezničkom sloju
- Fizički i poveznički sloj važni kada se povezuju uređaji
- Aplikacijski sloj važan kada se izgrađuje, postavlja, podešava aplikacija
- Cijeli protokolni stogovi koji odstupaju od Internetskog modela
- Posebne prilagodbe za jednostavne uređaje i na drugim slojevima
  - 6LoWPAN ("*IPv6 over Low-Power Wireless Personal Area Networks*")
- U *ad hoc* mrežama s više bežičnih skokova, npr. bežičnim senzorskim mreža, posebni protokoli usmjeravanja paketa kroz te mreže

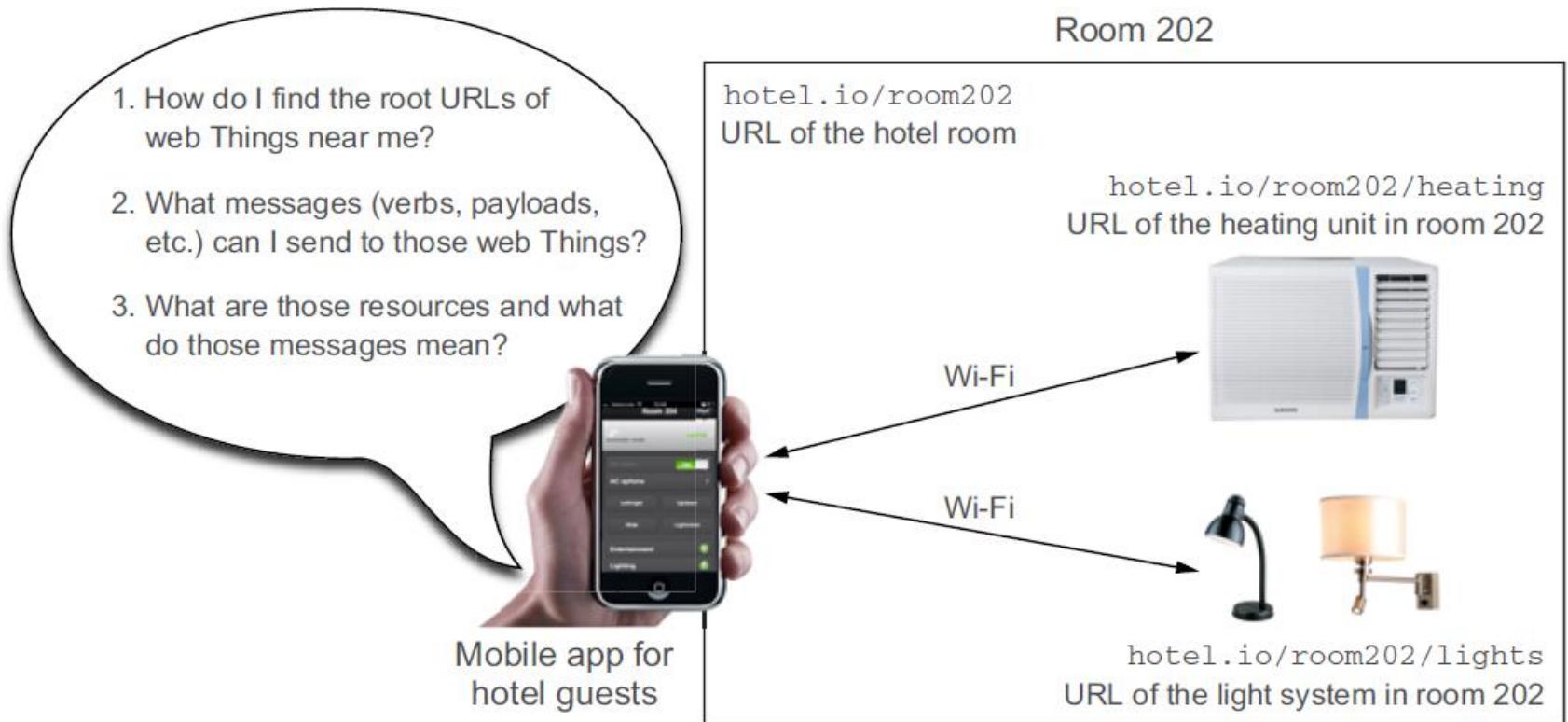
# Opisivanje i pretraživanje weba stvari

Darko Andročec

- Potreban je jedinstveni podatkovni model tako da stvari mogu izložite svoje metapodatke korištenjem samo web standarda i najboljih praksi
- Metapodaci znače opis web stvari, uključujući URL, naziv, trenutnu lokaciju i status, te usluge koje nudi, kao što su senzori, aktuatori, naredbe i svojstva.
- Prvo, ovo je korisno za otkrivanje web stvari dok se spajaju na lokalnu mrežu ili na web.
- Drugo, omogućuje aplikacijama, uslugama i drugim web stvarima traženje i pronalaženje novih uređaja bez instaliranja upravljačkog programa za tu stvar.

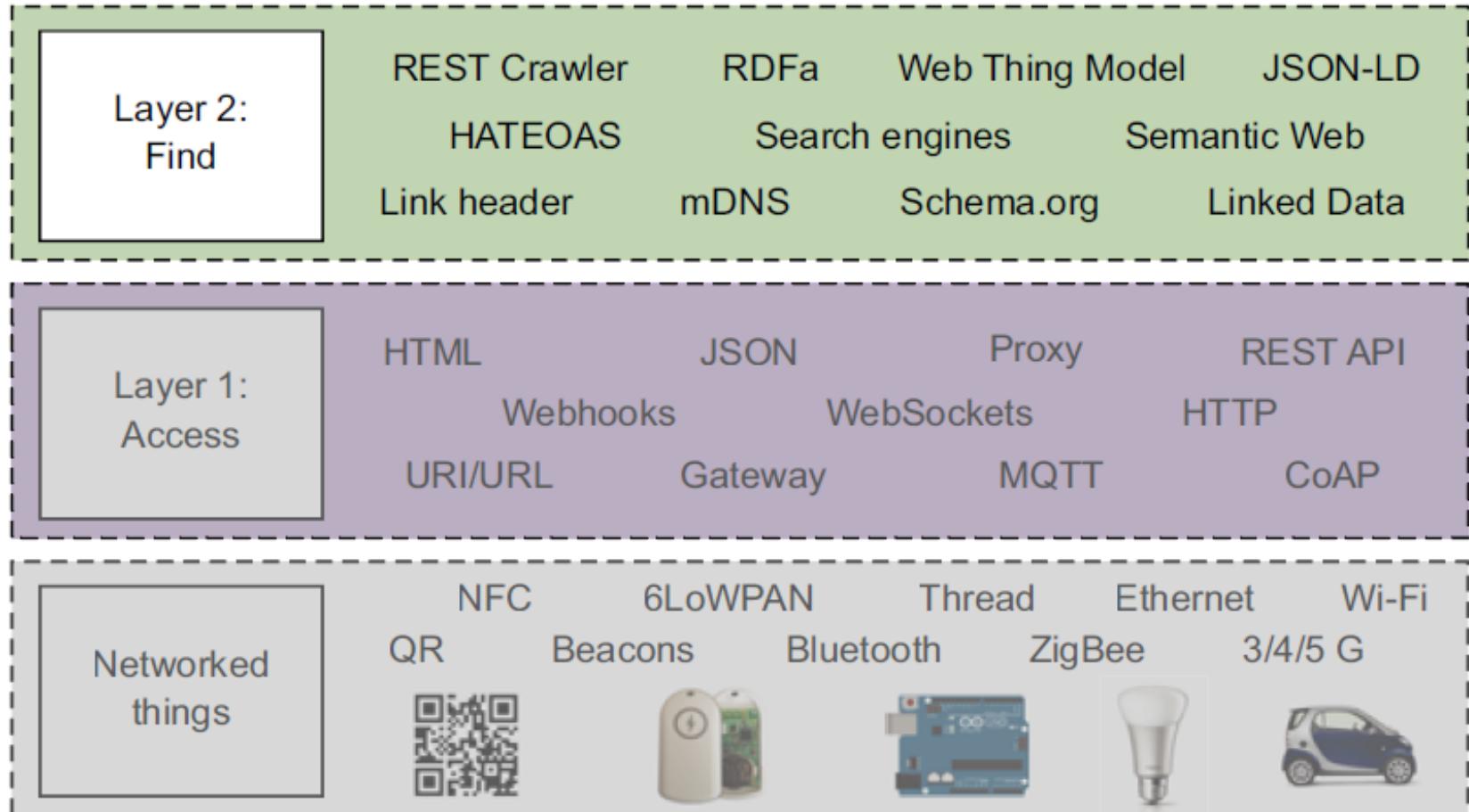
# Problem pronalaženja

- Kako znamo kamo poslati zahtjeve, kao što je korijenski URL i/ili resursi weba stvari?
- Kako znamo koje zahtjeve poslati i kako; na primjer, glagoli i format korisnih sadržaja (engl. *payloads*)?
- Kako znamo značenje zahtjeva koje šaljemo i odgovora koje dobivamo, odnosno semantiku?



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

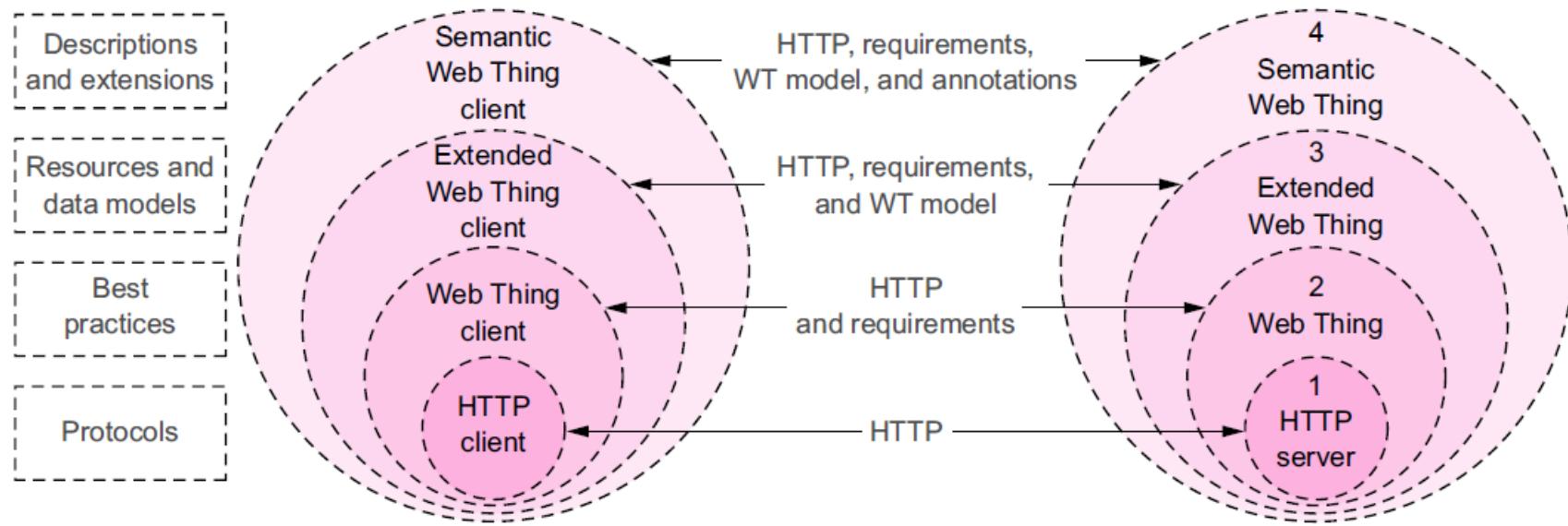
# Sloj za pretraživanje web stvari



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Otkrivanje u mreži – različiti protokoli za otkrivanje: multicast Domain Name System (mDNS), Digital Living Network Alliance (DLNA), Universal Plug and Play (UPnP)
- Mrežno otkrivanje na webu – web crawling stranica – gledaju se svi linkovi i otkrivaju sve još neindeksirane stranice. Kako to napraviti za web stvari?

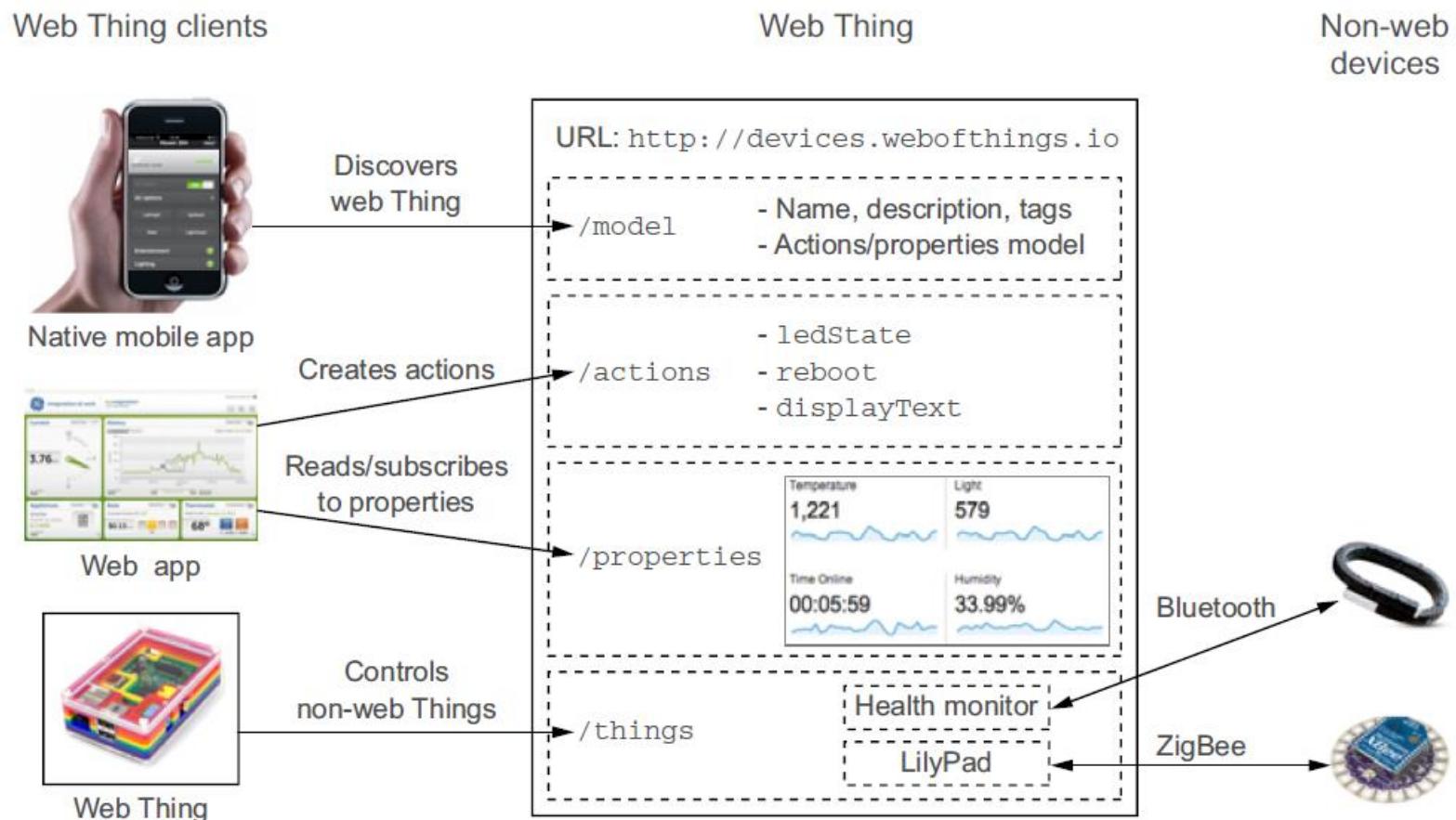
# Opisivanje web stvari



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Nakon što pronađemo web stvar i shvatimo njenu strukturu API-ja, i dalje nam treba metoda za opisivanje što taj uređaj jest i radi.
- Drugim riječima, trebamo konceptualni model web stvari koji može opisati resurse weba stvari korištenjem skupa dobro poznatih koncepata.
- Primjer: <http://gateway.webofthings.io/>
- Web stvari je digitalna reprezentacija fizičkog objekta

# Resursi web stvari



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

## Metapodaci

- U modelu weba stvari, sve web stvari trebaju imati pridružene metapodatke koji ih pobliže opisuju
- Ovo je skup osnovnih polja o webu stvari, uključujući njene identifikatore, naziv, opis i oznake, kao i skup resursa koje ima, kao što su akcije i svojstva.

- Web of Things (WoT) Thing Description -  
<https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/>
- Web of Things (WoT) Discovery -  
<https://www.w3.org/TR/wot-discovery/>

- Linked data
- RDF
- RDFa – jednostavnija verzija RDF-a dizajnirana i za računala i za ljude



# Platforme i softverski alati za prikupljanje podataka sa različitih senzora

Darko Andročec

- Tehnologije senzora koriste se za mnoge primjene, npr. robotika, vozila, vojne svrhe, pametne kuće i zgrade itd.
- Senzori su danas većinom bežični – portabilnost i jednostavnija instalacija
- Bežični senzor je u biti uređaj opremljen senzorskim hardverom i primopredajnim modulom
- Senzorski hardver omogućuje mu da osjeti podatke iz svoje okoline; modul primopredajnika omogućuje senzorskem uređaju primanje ili slanje podataka u obliku paketa.

- Wireless Sensor Networks (WSNs)
- WSN se sastoji od više senzorskih čvorova koji tvore mrežu.
- Svaki čvor u WSN-u očitava putem vlastitog hardvera i prenosi očitane informacije preko bežičnog medija do odredišta ili čvora odvodnika putem rute s jednim ili više skokova (engl. single- or multi-hop route).
- Konvencionalni WSN-ovi teško se nose s upravljanjem sve veće količine podataka sa senzora, pa se često za postavljanje WSN-ova koristi i oblak

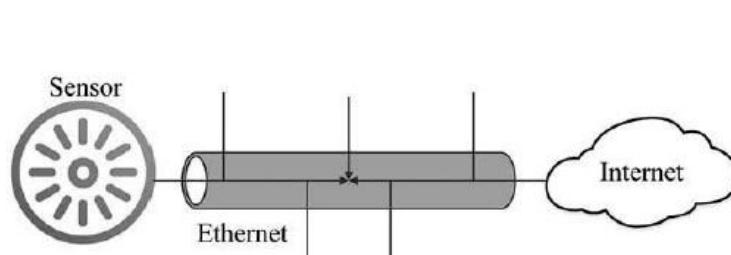
## Karakteristike WSN-ova (1)

- Hardver – veličina senzorskih čvorova varira od nekoliko kubnih cm do nekoliko kubnih dm
- Komunikacija – čvor ima komunikacijski primopredajnik koji podržava jedan od standarda → IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (ZigBee), IEEE 802.16 (WiMax) itd.
- Usmjeravanje - Senzorski čvorovi generiraju pakete podataka na temelju očitanih informacija. Paketi podataka se zatim usmjeravaju prema odvodniku(engl. sink) izravno ili putem rute s više skokova pomoću drugih čvorova, na temelju dizajna topologije mreže.

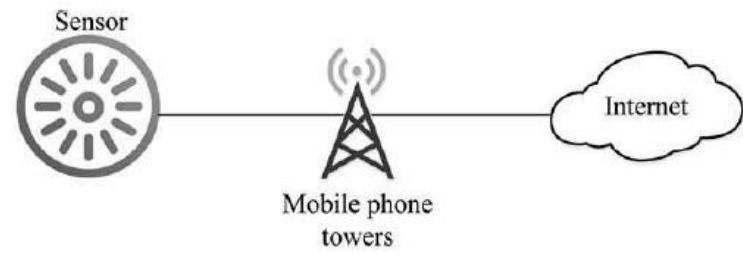
## Karakteristike WSN-ova (2)

- Interni softver – senzori uglavnom sadržavaju operacijski sustav otvorenog koda, npr. TinyOS - TinyOS biblioteke uključuju upravljačke programe senzora i alate koji omogućuju prikupljanje podataka, pohranjivanje i upravljanje napajanjem
- Obradovanje podataka - uključuje obradu podataka, kalibraciju, fuziju, agregaciju i donošenje odluka.
- Upravljanje mrežom - pojedinosti o mrežnoj arhitekturi i komunikacijskim standardima, toleranciji na pogreške mreže, učinkovitosti resursa i komunikacijskim frekvencijama

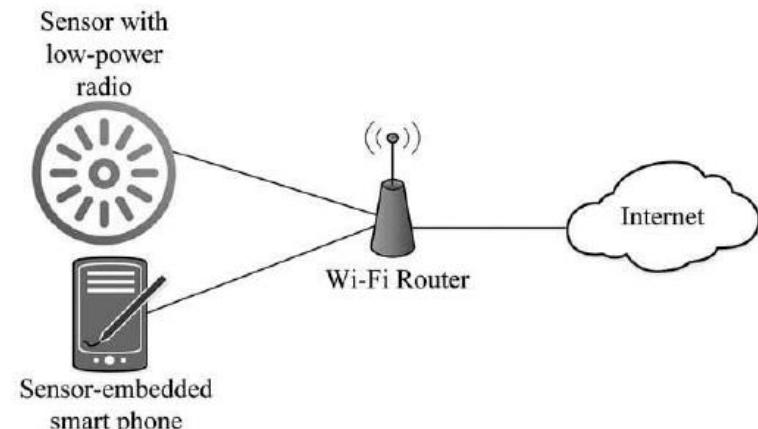
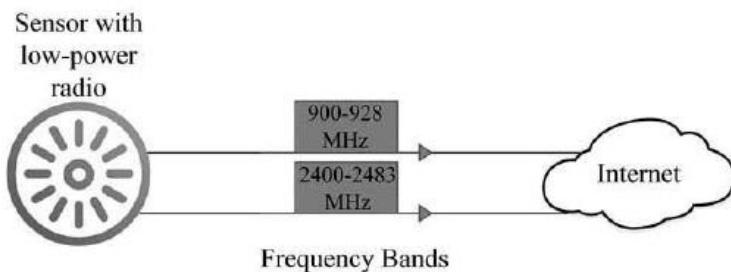
# WSN-ovi i internet



(a) Before 1980s



(b) Between 1980s and 1990s



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

# Tipovi senzora

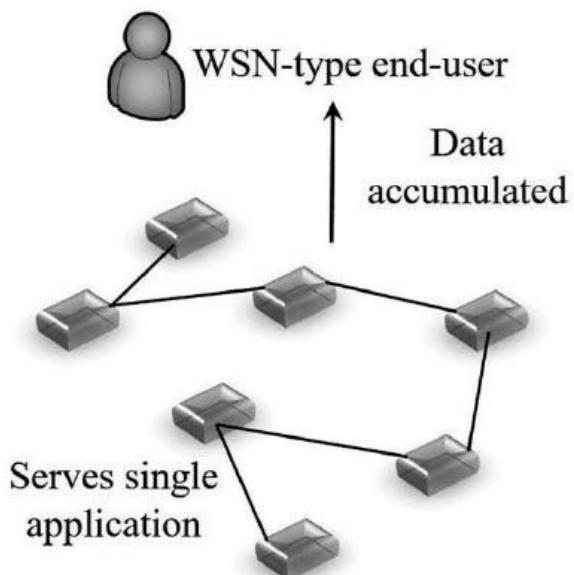
- Infracrveni senzori
- Senzori vida
- Senzori razine
- Senzori protoka
- Senzori plina
- Senzori pokreta
- Senzori dodira ekrana
- Senzori za praćenje okoliša

- Ovi senzori opremljeni su inteligencijom za rješavanje složenosti računanja i donošenja odluka unutar sebe, čime omogućuju sustavima domaćina da uživaju u apstraktnosti i jednostavnosti.
- Stoga je dizajn pametnog senzora poboljšan kako bi podržao napredne funkcije.
- Prednosti pametnog senzora uključuju bolje održavanje, kraće vrijeme zastoja, veću pouzdanost, otpornost na pogreške, manju težinu i bolju arhitekturu sustava.

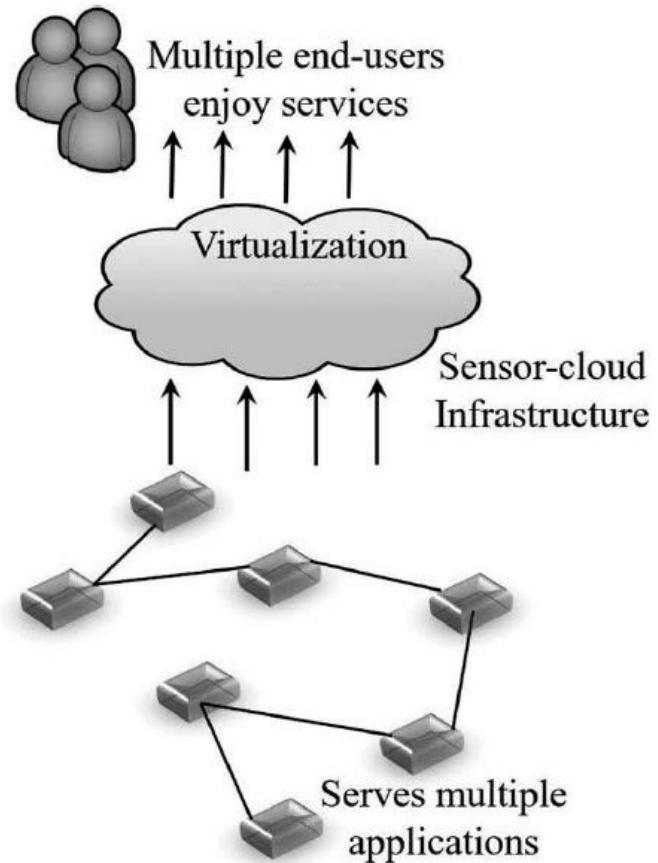
- Oblak pruža platformu za analitiku podataka i podatkovno spremište za mnoge WSN-ove
- Mobilne senzorske mreže na oblaku
- Green Mobile Cloud Computing okvir
- Računalstvo u oblaku dobro se integrira s poljoprivrednim senzorskim mrežama za pružanje usluga poljoprivrednicima po vrlo niskoj cijeni
- Primjena u zdravstvu

- Infrastruktura senzorskog oblaka, koja je u biti proširenje konvencionalnog računalstva u oblaku, temelji se na principu virtualizacije fizičkih senzorskih čvorova, stvarajući tako moćnu infrastrukturu koja povezuje fizički i kibernetički svijet.
- Infrastruktura senzorskog oblaka definirana je kao jedinstvena platforma za pohranu podataka senzora, vizualizaciju i daljinsko upravljanje koja iskorištava snažne tehnologije računarstva u oblaku kako bi pružila izvrsnu skalabilnost podataka, brzu vizualizaciju i korisničku programabilnu analizu.
- Temeljni postupci dobivanja neobrađenih očitanih podataka te njihove obrade i združivanja u značajne informacije potpuno su apstrahirani za krajnje korisnike.

# Analiza neovisnosti topologije



(a) Wireless Sensor Network



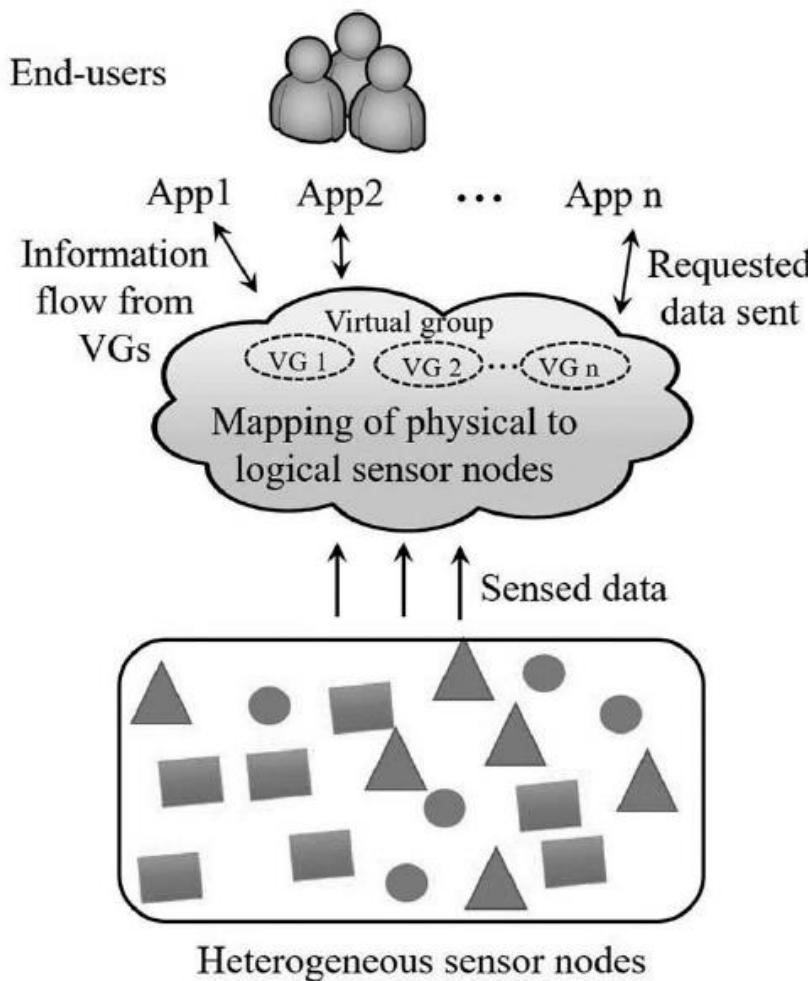
(b) Sensor-cloud

Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

- Virtualizacija fizičkih senzorskih čvorova omogućuje krajnjim korisnicima da zamisle senzore u obliku usluge, općenito poznate kao Senzori-kao-usluga (Se-aaS).
- Za razliku od tradicionalnih WSN-ova, Se-aaS stvara novu percepciju i razbija konvencionalni način zamišljanja senzorskih čvorova, ne samo kao tipičnog hardvera, već kao jednostavne dostupne usluge, baš poput vode ili struje.
- Se-aaS je više ekonomičan nego klasičan WSN

- Krajnji korisnici - osoba (ili organizacija) koja posjeduje vlastite aplikacije koje se trebaju puniti senzorskim podacima iz fizičkih senzorskih mreža – skalabilnost, plaćanje prema korištenju Se-aaS-a
- Vlasnik senzora – tvrtka koja kupuje fizičke senzorske uređaje i iznajmljuje ih
- Administrator senzorskog oblaka - upravlja i kontrolira sve aktivnosti obrade podataka unutar infrastrukture oblaka

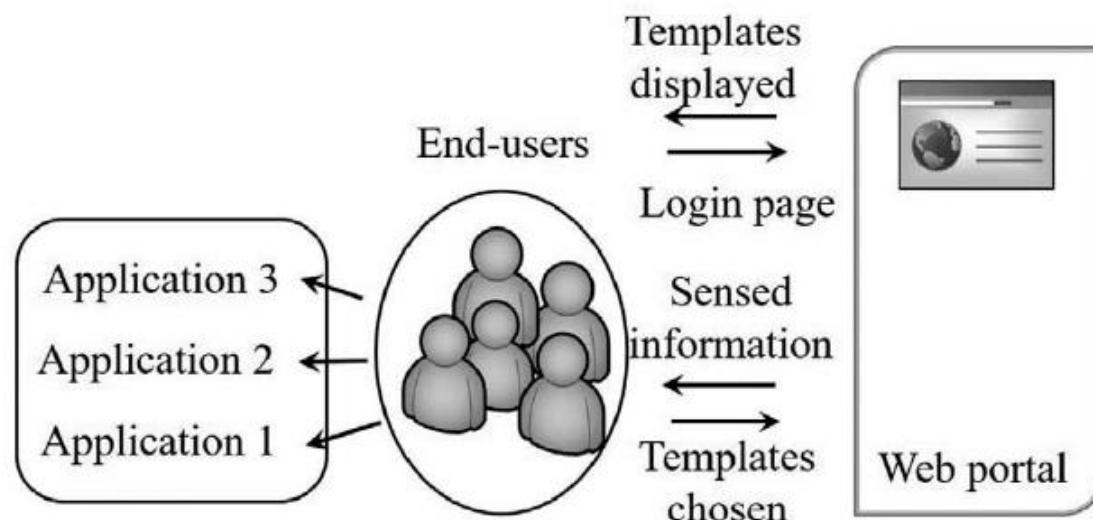
# Arhitektura senzorskog oblaka



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things [4]

# Pogledi na senzorski oblak (1)

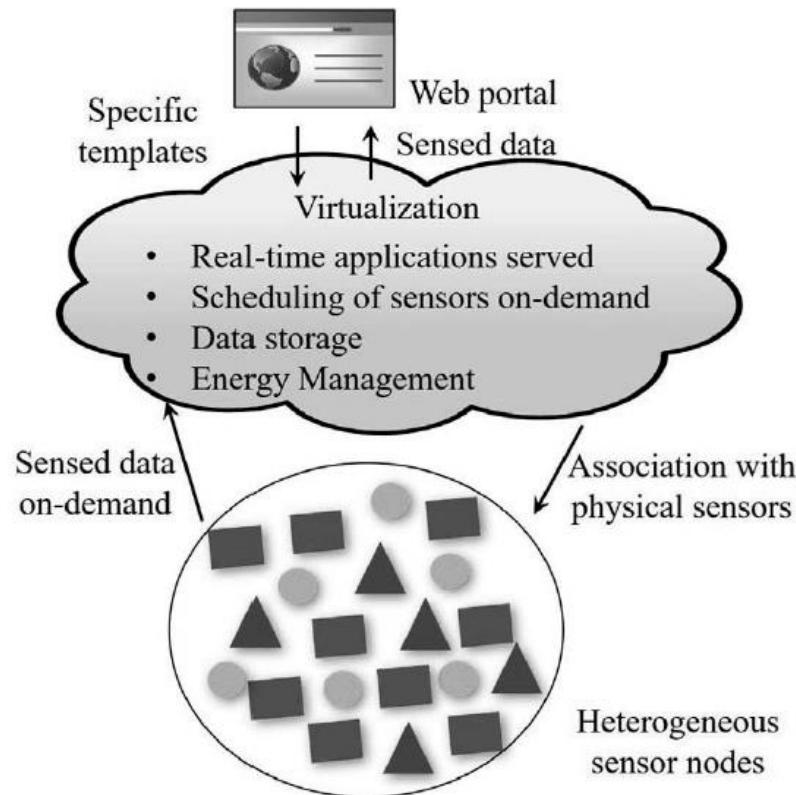
- Pogled korisnika/organizacije (logički pogled) – odgovara perspektivi krajnjeg korisnika senzorskog oblaka.



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

# Pogledi na senzorski oblak (2)

- Algoritamski pogled (realni pogled) - virtualizacija



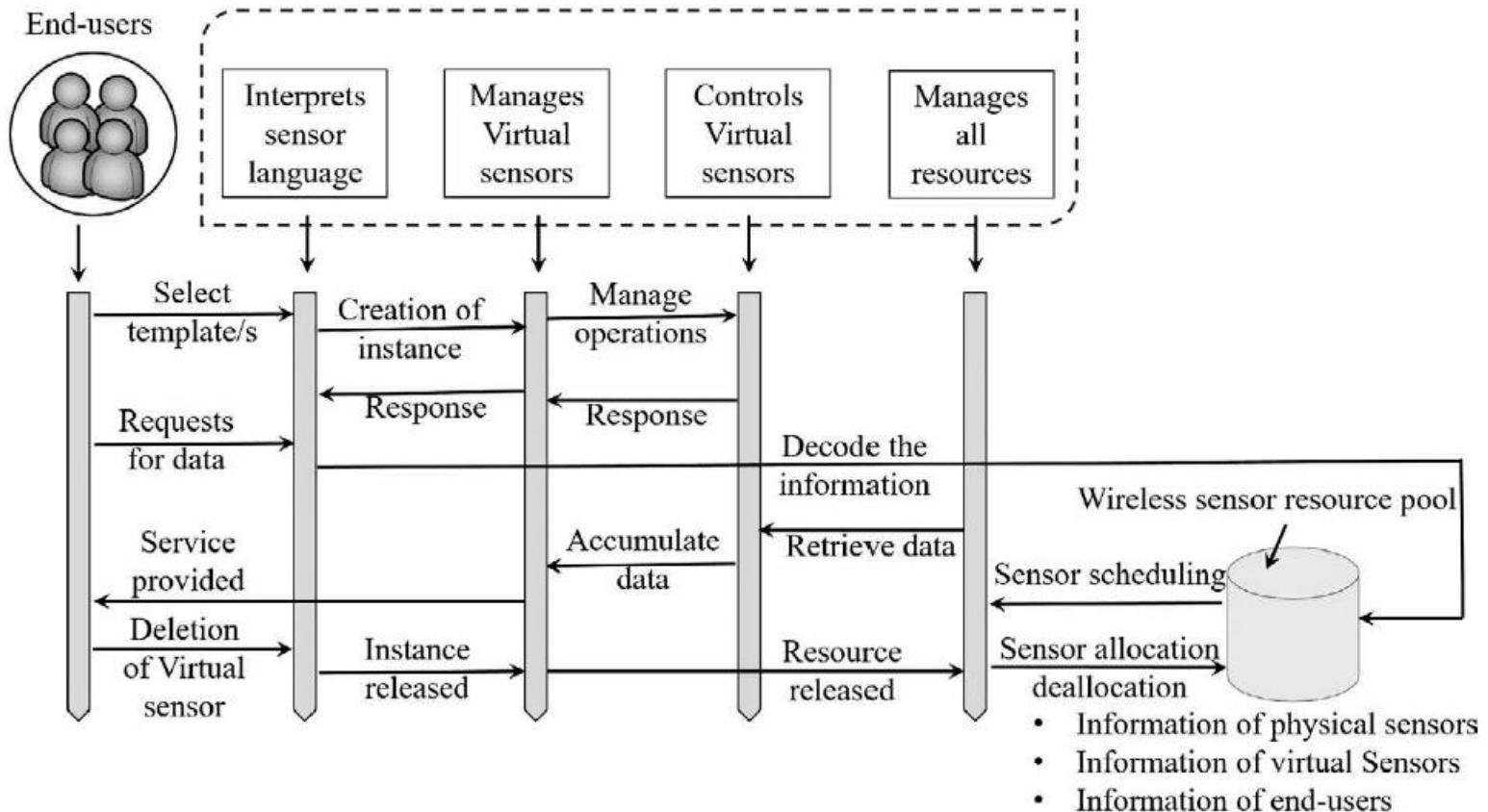
Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

- = je logično ujedinjenje nekoliko nesusjednih WSN-ova koji koordiniraju i surađuju kako bi opsluživali aplikaciju sa zahtjevima geo-distribuirane obrade podataka
- Mrežni korisnik potpuno je apstrahiran od složenosti povezivanja i sinkronizacije više geo-distribuiranih WSN-ova za posluživanje jedne aplikacije.
- Naprotiv, u senzorskom oblaku virtualizacija uključuje proces grupiranja čvorova koji bi sudjelovali i u prikupljanju senzorskih podataka i u komunikaciji.

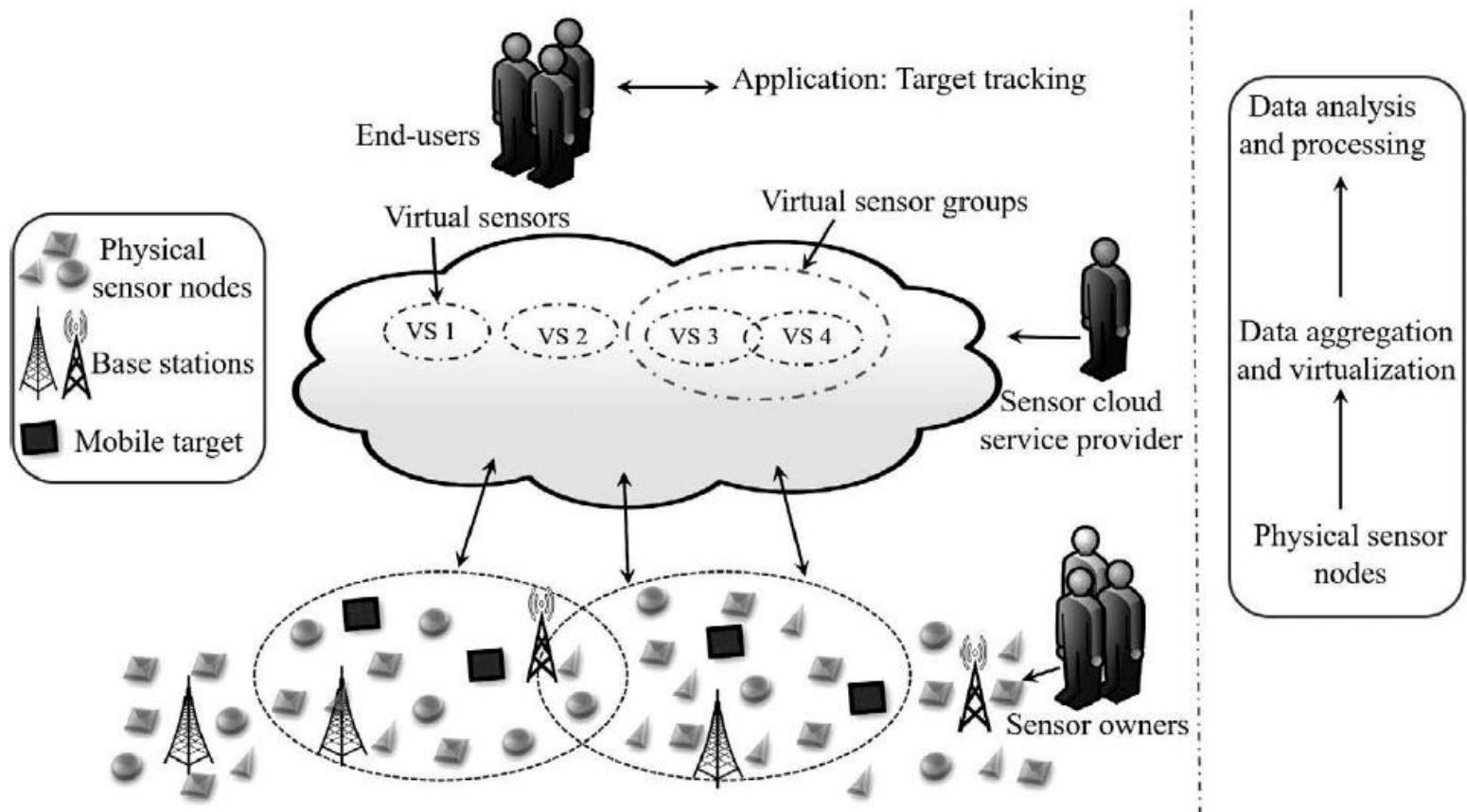
- Virtualizacija fizičkih senzora
- Virtualni senzori - skup fizičkih senzora iz kojih se podaci dobivaju i agregiraju
- Logička slika nekoliko fizičkih senzora koji ga čine

- Konfiguracija jedan naprema više (engl. one-to-many)  
– jedan fizički senzor predstavlja se sa više virtualnih senzora
- Konfiguracija više prema jedan (engl. many-to-one) – više fizičkih senzora doprinosi jednom virtualnom senzoru – npr. informacije iz veše geografskih područja
- Konfiguracija više-više (engl. many-to-many) – kombinacija prethodne dvije konfiguracije
- Izvedena konfiguracija – koristi principe gore navedenih triju konfiguracija sa razlikom da se radi o senzorima različitih tipova

- Nimbit - open-source platforma za obradu podataka koja omogućuje snimanje i pohranjivanje podataka senzora u oblaku
- Pachube Platform – jedan od prvih online pružatelja baza podataka
- iDigi - PaaS platforma koja pruža usluge stroj-stroj za sredstva uređaja organizacije
- ThingSpeak - aplikacija otvorenog koda koja omogućuje prikupljanje i pohranjivanje podataka senzora putem LAN-a ili HTTP-a preko interneta



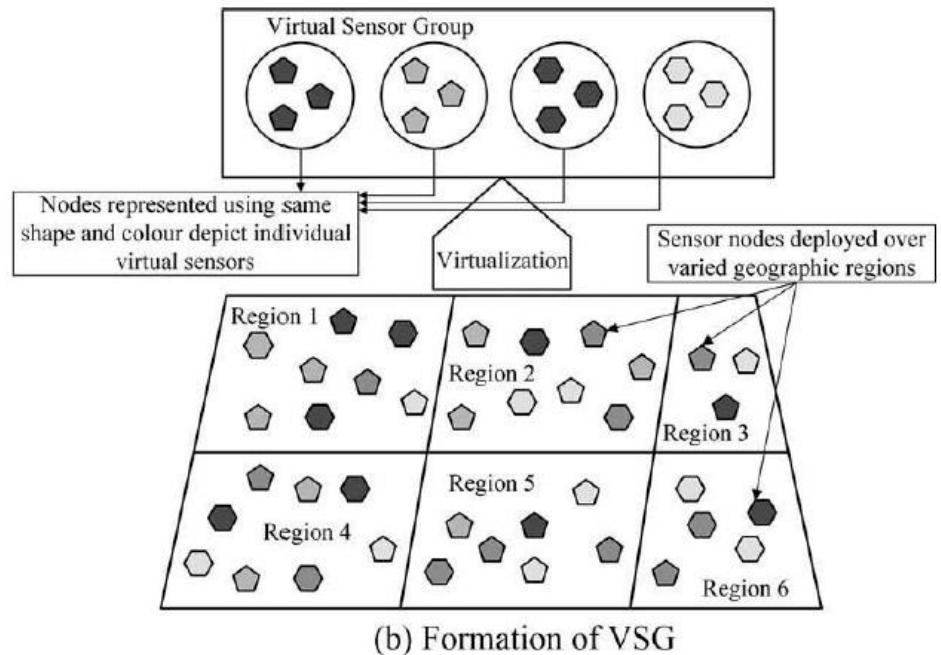
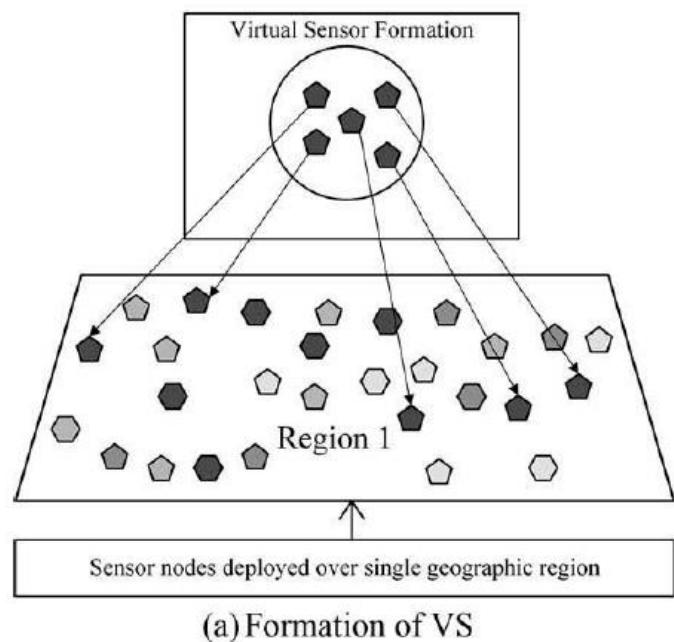
# Studija slučaja: otkrivanje upada



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things  
22

- Fizički senzori se alociraju na osnovu zahtjeva od strane aplikacije
- Senzori se logički grupiraju u virtualne senzore, a virtualni senzori se grupiraju u grupe virtualnih senzora (engl. Virtual Sensor Groups)
- Obrazloženje ta optimalnu kompoziciju virtualnih senzora potkrijepljena je činjenicom da je životni vijek senzorskih čvorova pokretanih baterijama ograničen, nakon čega se čvorovi moraju napuniti ili zamijeniti.

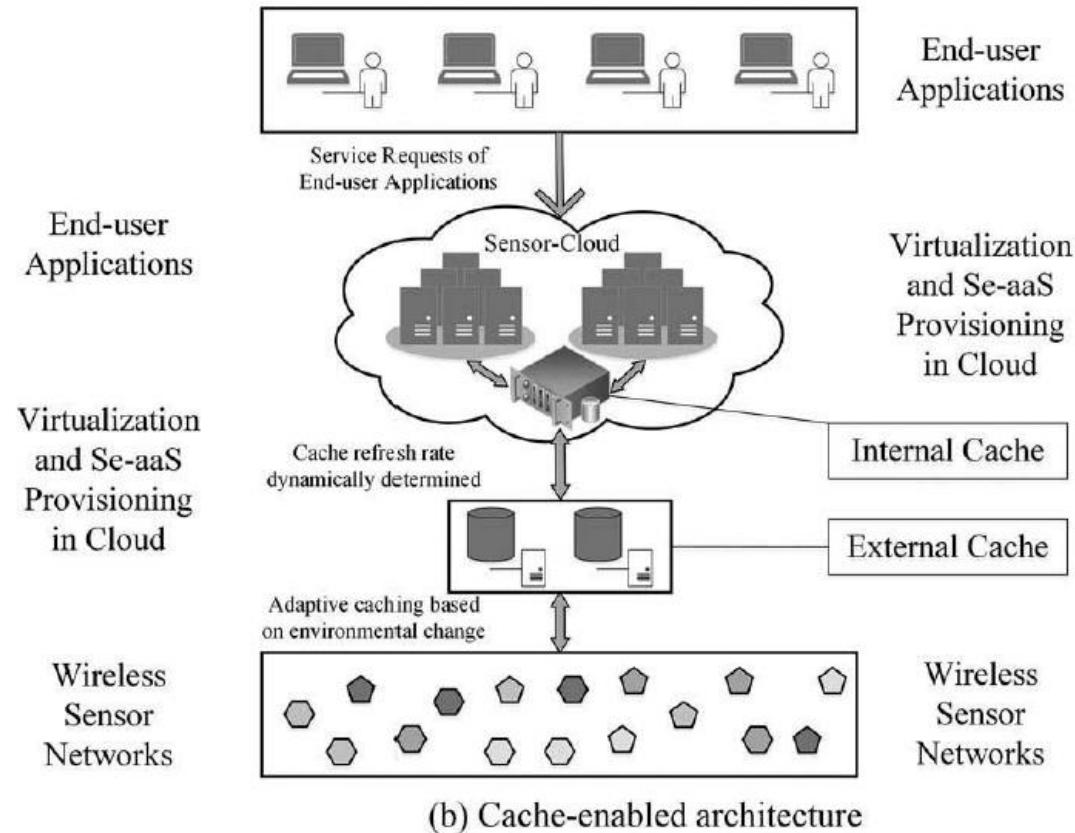
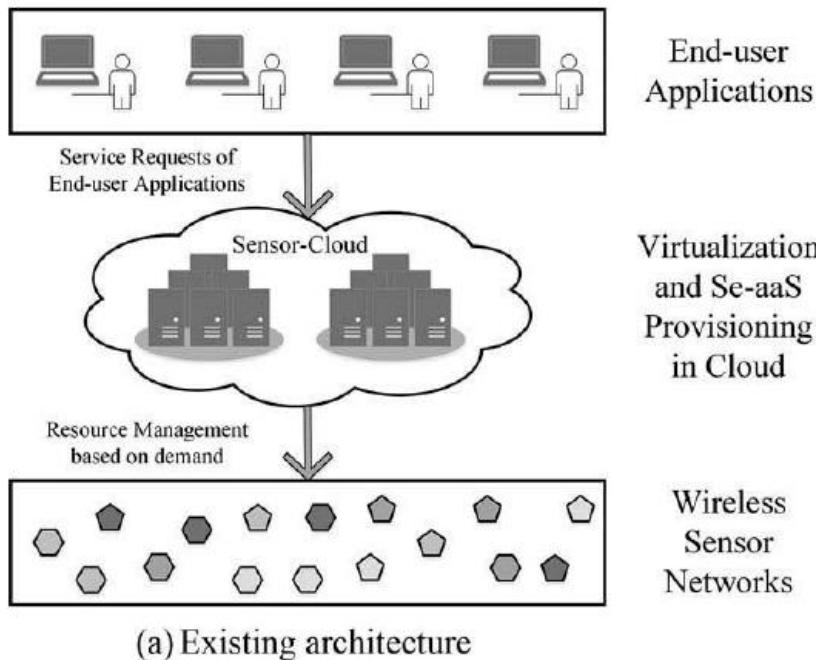
# Virtualizacija fizičkih senzora



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

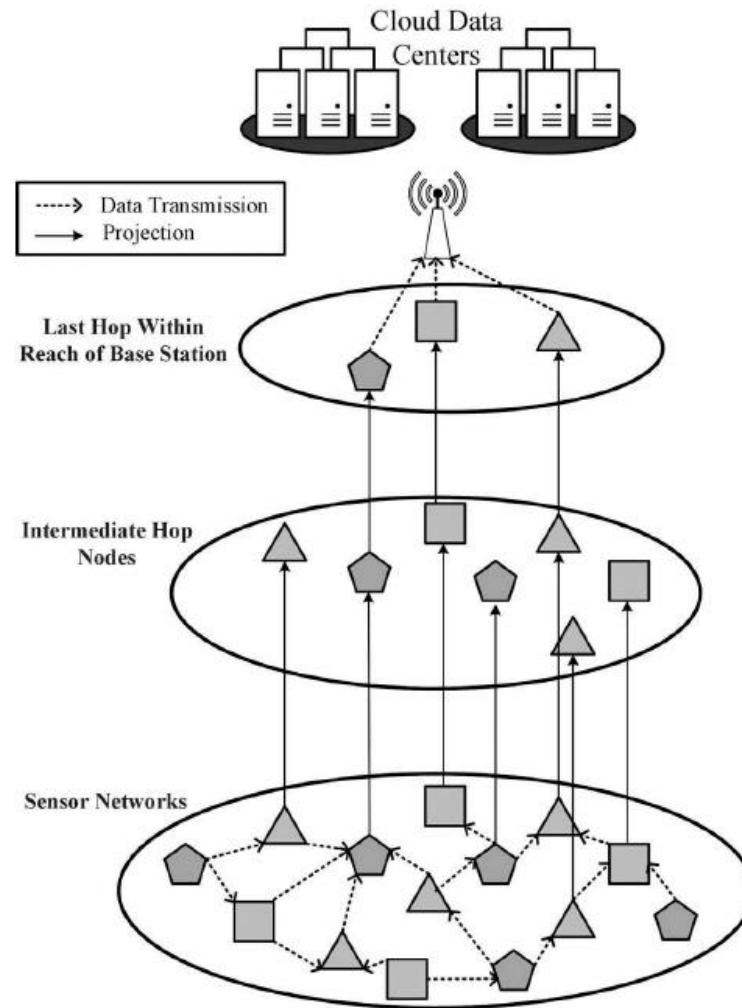
- Kad god aplikaciju treba poslužiti sa Se-aaS-om, odgovarajući fizički senzori se identificiraju i aktiviraju, a podaci se naknadno dohvaćaju iz senzora
- Ako aplikacija ima velike zahtjeve, podaci se izvlače češće iz senzorskih mreža od onih s malim zahtjevima.
- Senzor kamere – često idu podaci – smisleno je imati caching podatakav - poboljšava potrošnju energije i životni vijek mreže te također čuva točnost informacija

# Arhitektura cachiranja podataka u senzorskom oblaku



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

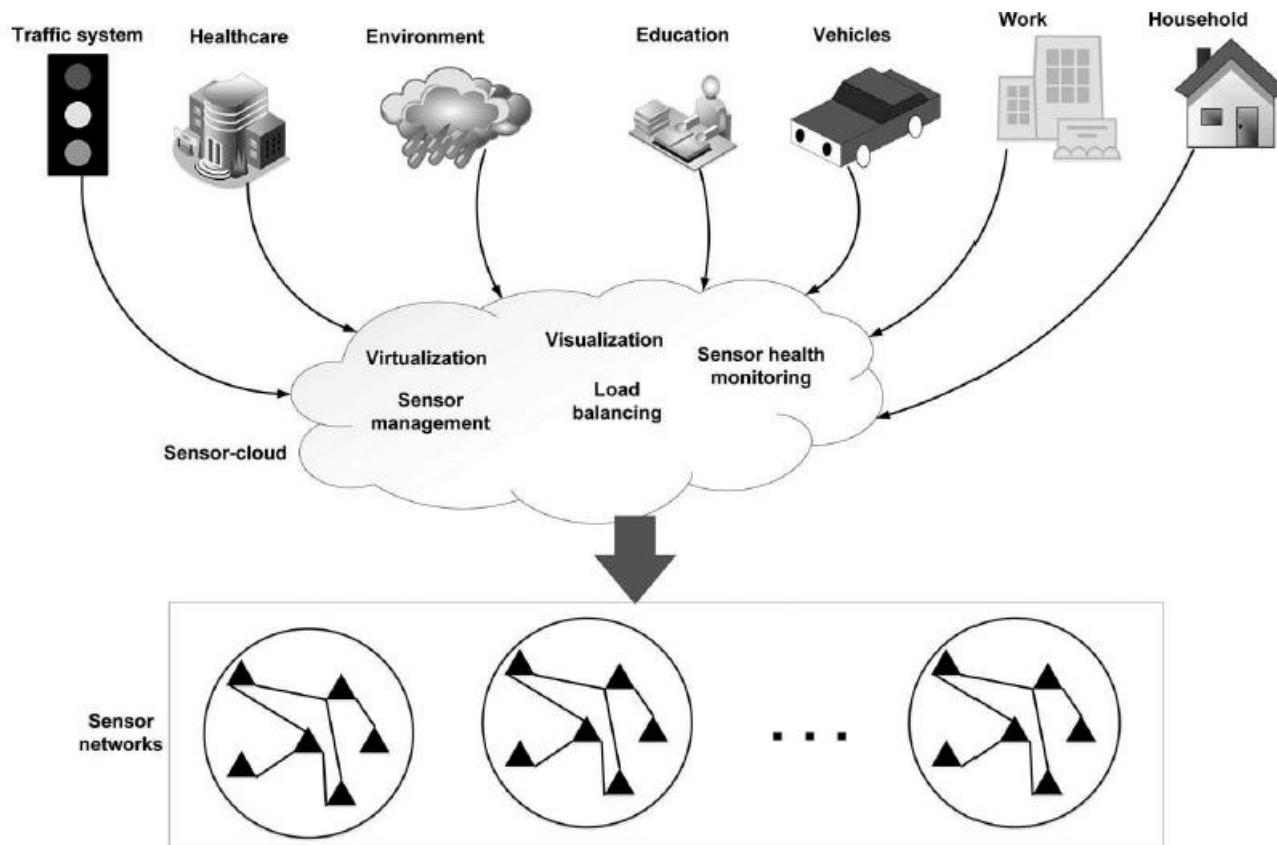
# Prijenos podataka iz više mreža virtualnih senzora



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things 27

- Budući da je senzorski oblak proširenje konvencionalne infrastrukture računalstva u oblaku, on se pridržava modela plaćanja prema korištenju (engl. pay-as-you-go)
- Krajnjim korisnicima također se naplaćuju samo usluge koje su im ponuđene u smislu podataka senzora, infrastrukture oblaka i resursa.
- Očekuje se da će politike cijena unutar senzorskog oblaka biti dinamične i prilagodljive promjenama u potražnji korisnika.
- Cilj je optimizirati cijenu koja se naplaćuje od krajnjih korisnika i profitnu maržu pružatelja usluga u oblaku i vlasnika senzora.

# Pogled na senzorski oblak



Iz: Sensors, Cloud, and Fog: The Enabling Technologies for the Internet of Things

- Analiza - Analitika senzorskih podataka ključna je komponenta i za senzorski oblak i za IoT.
- Kolaboracija - U tom smislu, senzorski oblak još uvijek se istražuje; fokus je na integraciji izrazito heterogenih senzora u hardveru, komunikaciji i dometu osjeta, komunikacijskim protokolima i standardima.
- Vizualizacija – API-ji za krajnje korisnike za upravljanje senzorima



# Interoperabilnost servisa na oblaku

Darko Andročec

- IEEE definicija: „sposobnost dva ili više sustava ili komponenti da razmjenjuju informacije i da koriste informacije koje su razmijenjene”
- „Sposobnost zbirke entiteta koji komuniciraju da (a) dijele određene informacije i (b) rade na tim informacijama u skladu s dogovorenom operativnom semantikom” - Brownsword et al.
- „Interoperabilnost podrazumijeva da sustavi mogu međusobno komunicirati (tj. razmjenjivati poruke), čitati i razumjeti međusobne poruke i dijeliti ista očekivanja o učinku razmjene poruka” - Pokraev et al.
- „Sposobnost da sustav komunicira s drugim sustavom i da koristi funkcionalnost drugog sustava” - Vernadat

- Interoperabilnost je najkritičnije pitanje za tvrtke koje koriste podatke iz različitih informacijskih sustava.
- Semantička interoperabilnost postoji na razini znanja i koristi se za premošćivanje semantičkih konflikata zbog razlika u značenjima, perspektivama i pretpostavkama
- Sintaktička interoperabilnost je interoperabilnost na razini aplikacije koja ima za cilj suradnju između različitih softverskih komponenti s različitim implementacijskim jezicima i razvojnim platformama

# Problemi interoperabilnosti (1)

- Europski okvir interoperabilnosti (EIF) (63) ima za cilj interoperabilnost europskih javnih usluga i identificira četiri razine interoperabilnosti: pravnu (zbog nekompatibilnosti zakonodavstva u različitim zemljama EU), organizacijsku (neusklađenost poslovnih procesa), semantičku (uzrokovana je sukobima značenja podatkovnih elemenata i formata razmijenjenih informacija) i tehničku interoperabilnost.

- Sheth i Kashyap
- nekompatibilnost definicije domene (atributi imaju različite definicije domene)
- nekompatibilnost definicije entiteta (deskriptori koji se koriste za isti entitet djelomično su kompatibilni)
- nekompatibilnost vrijednosti podataka (nedosljednost između povezanih podataka)
- nekompatibilnost razine apstrakcije (isti entitet predstavljen je na različitim razinama apstrakcije)
- shematsko odstupanje (podaci u jednoj bazi podataka odgovaraju elementima sheme u drugoj)

- Park i Ram
- Konflikti na razini podataka uključuju konflikte vrijednosti podataka (vrijednost podataka ima različito značenje u različitim bazama podataka), konflikte u prikazu podataka (kao što su različiti prikazi datuma i vremena), konflikte u jedinicama podataka (različite jedinice koriste se u različitim bazama podataka) i konflikte preciznosti podataka.
- Svi konflikti na razini podataka mogu se pojaviti na razini atributa ili na razini entiteta.
- Konflikti na razini sheme uključuju: konflikte imenovanja, probleme identifikatora entiteta, izomorfizme shema, konflikte generalizacije, konflikte agregacije i shematske nedosljednosti.

- Nagarajan i dr.
- nekompatibilnosti na razini atributa (različiti opisi se koriste za modeliranje sličnih atributa)
- nekompatibilnosti definicija entiteta (različiti opisi se koriste za modeliranje sličnih entiteta)
- nekompatibilnosti na razini apstrakcije - različite razine apstrakcije
- metode koje nedostaju, dodatna polja, polja koja nedostaju, nepodudaranja facet-a (različite vrste za ulazna ili izlazna polja) i nepodudaranja kardinalnosti (različiti zahtjevi kardinalnosti za polje)

# Razine interoperabilnosti

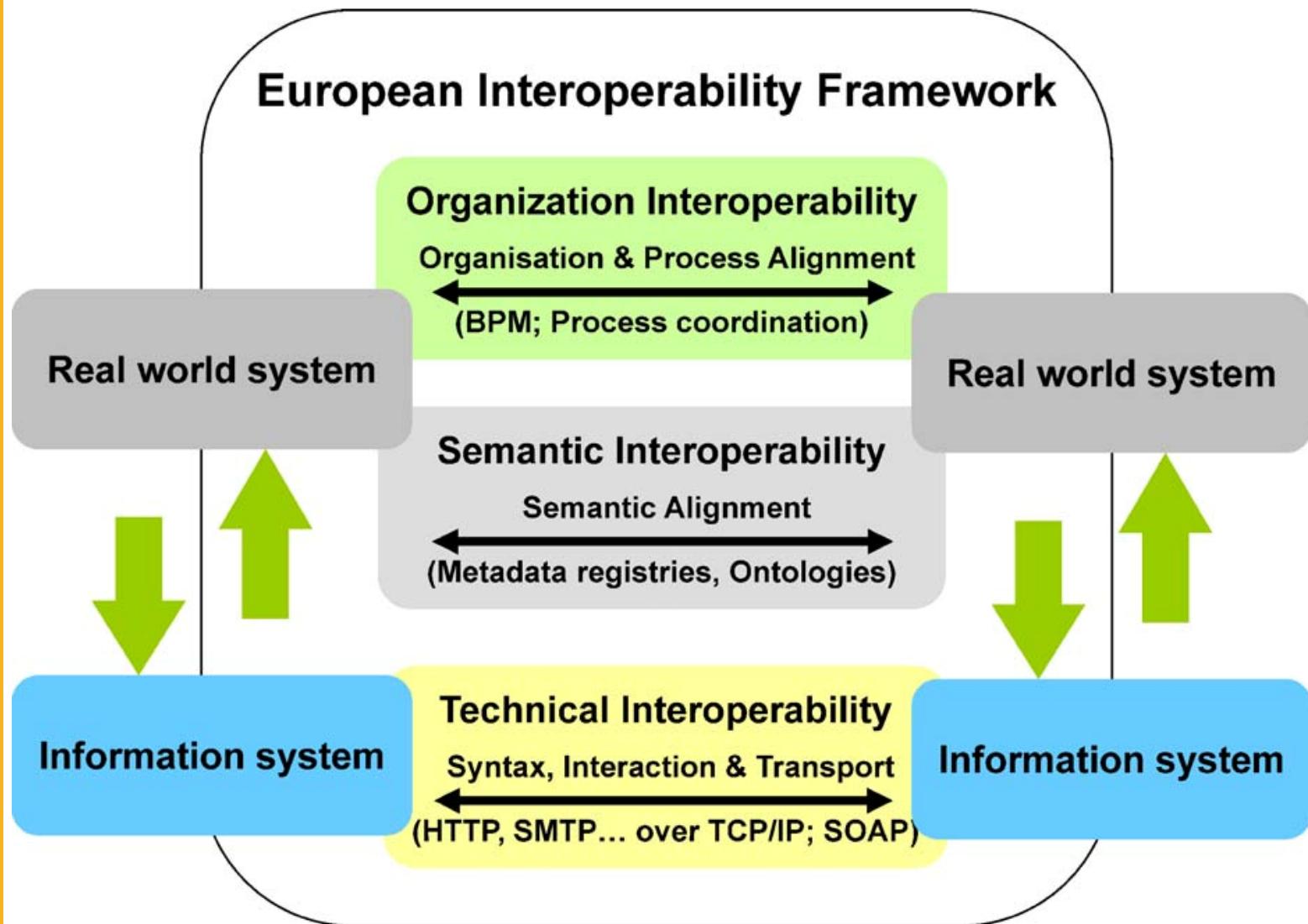
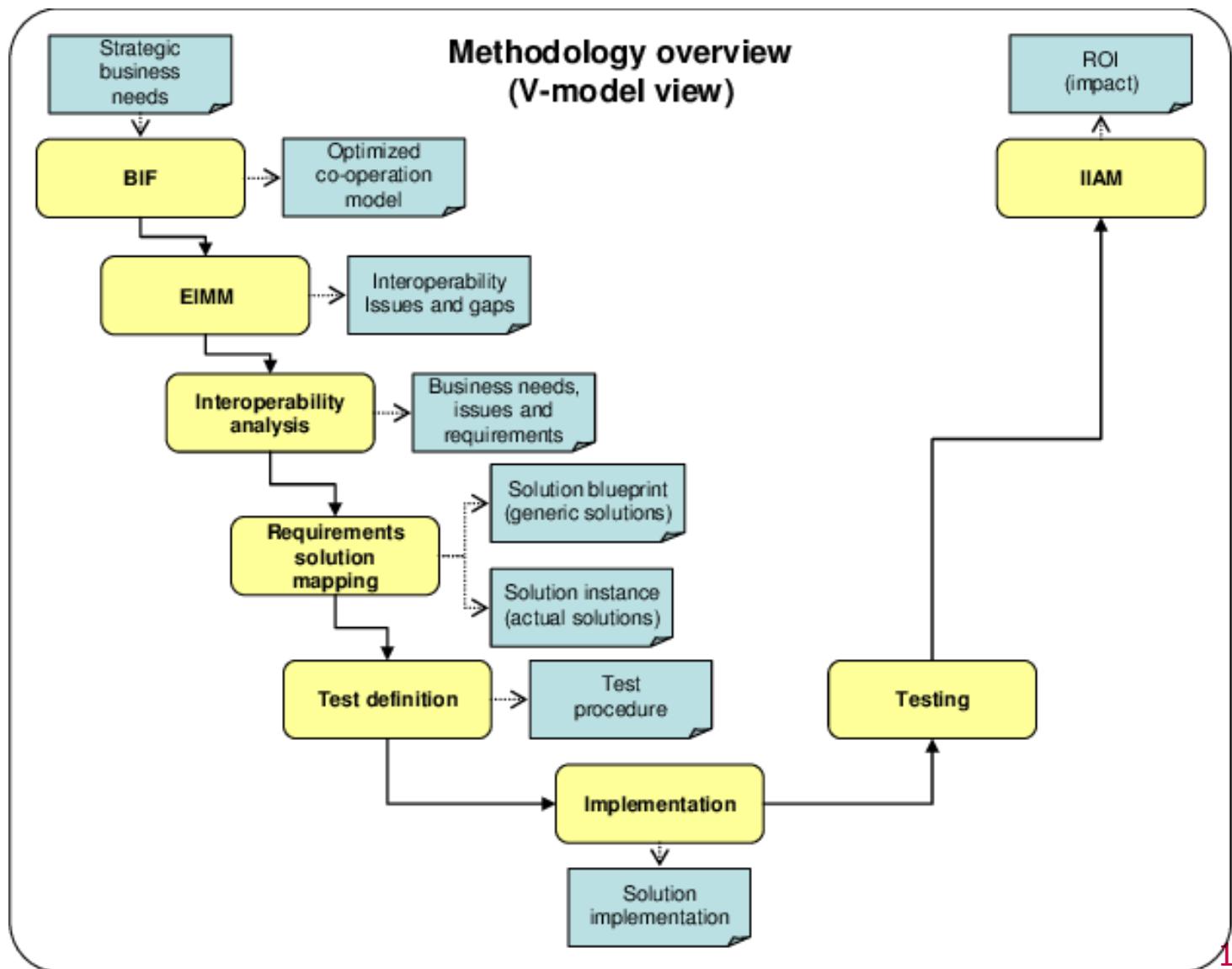


Fig. 1. European Interoperability Framework.

- Interoperabilnost je višedimenzionalni koncept koji se može promatrati iz više perspektiva.
- Razvijeni su okviri za interoperabilnost koji određuju skup zajedničkih elemenata kao što su rječnik, koncepti, načela, smjernice i preporuke
- ATHENA Interoperability Framework (AIF), IDEAS Interoperability Framework, LISI Reference Model, Enterprise Interoperability Framework, GridWise Interoperability Context-Setting Framework.

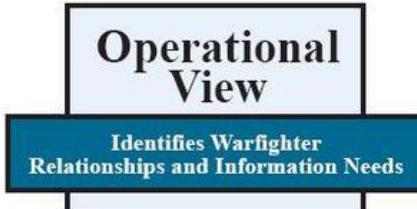
# ATHENA Interoperability Framework (AIF),



# LISI Reference Model

Discipline for detailing  
Node-to-Node  
Relationships

Specified Interoperability  
Level Required for  
Each Operational  
Needline



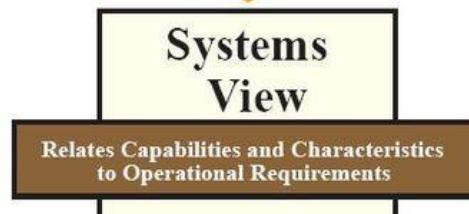
Nature of Operational Information Interaction	Corresponding Computing Environment Level	Code	Implications			
			P	A	I	D
Cross-Domain Interactive Manipulation	Universal	4	Enterprise Level	Interactive	Multiple Topologies	Enterprise Model
Shared Applications & Databases	Integrated	3	Domain Level	Groupware	World Wide Networks	Domain Model
Complex Media Exchange	Distributed	2	Program Level	Desktop Automation	Local Networks	Program Model
Simple Electronic Exchange	Connected	1	Local/Site Level	Standard System Drivers	Simple Connection	Local
Manual Gateway	Isolated	0	Access Control	N/A	Independent	Private

Discipline for creating  
Interoperability-focused  
Technical Profiles

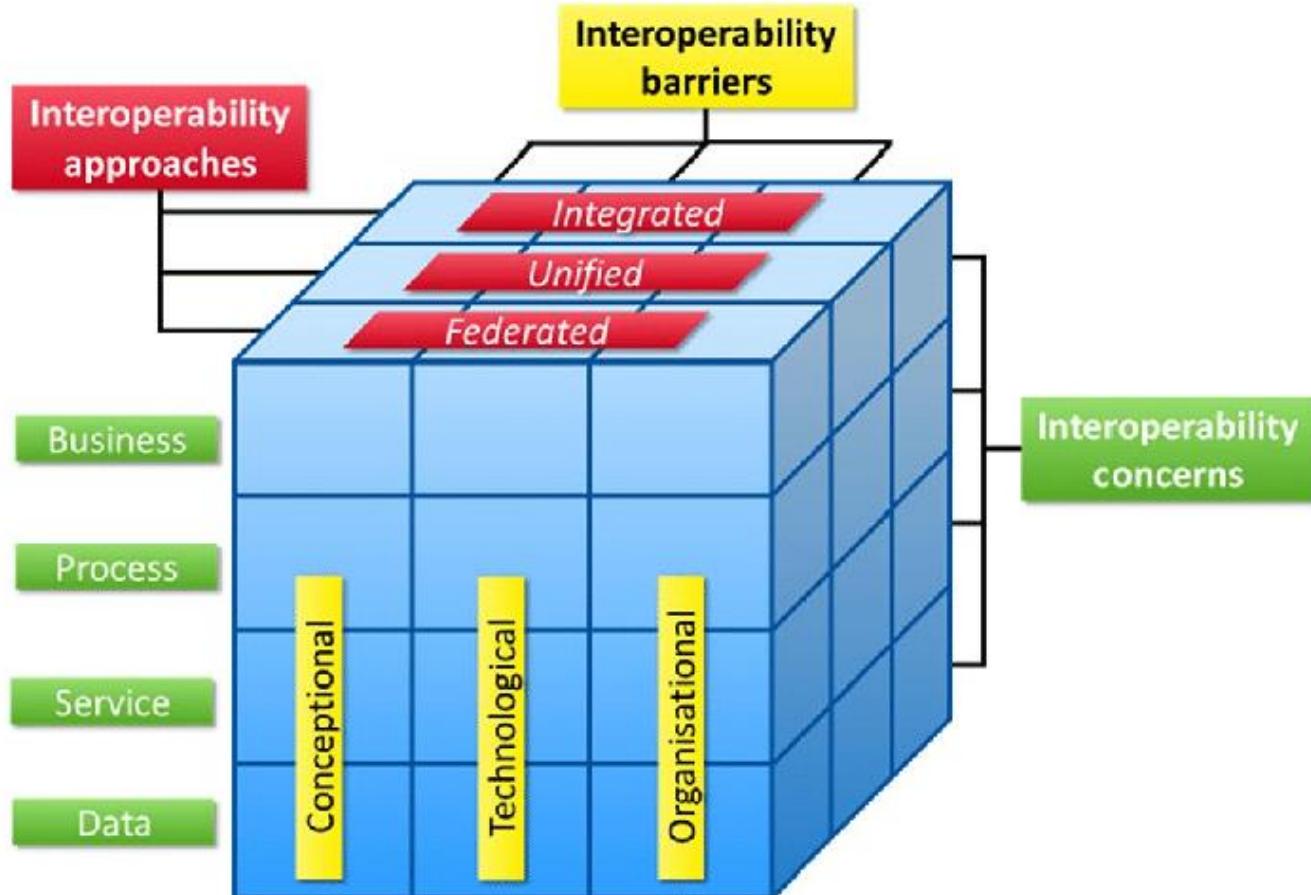
System  
Implementation  
Rules



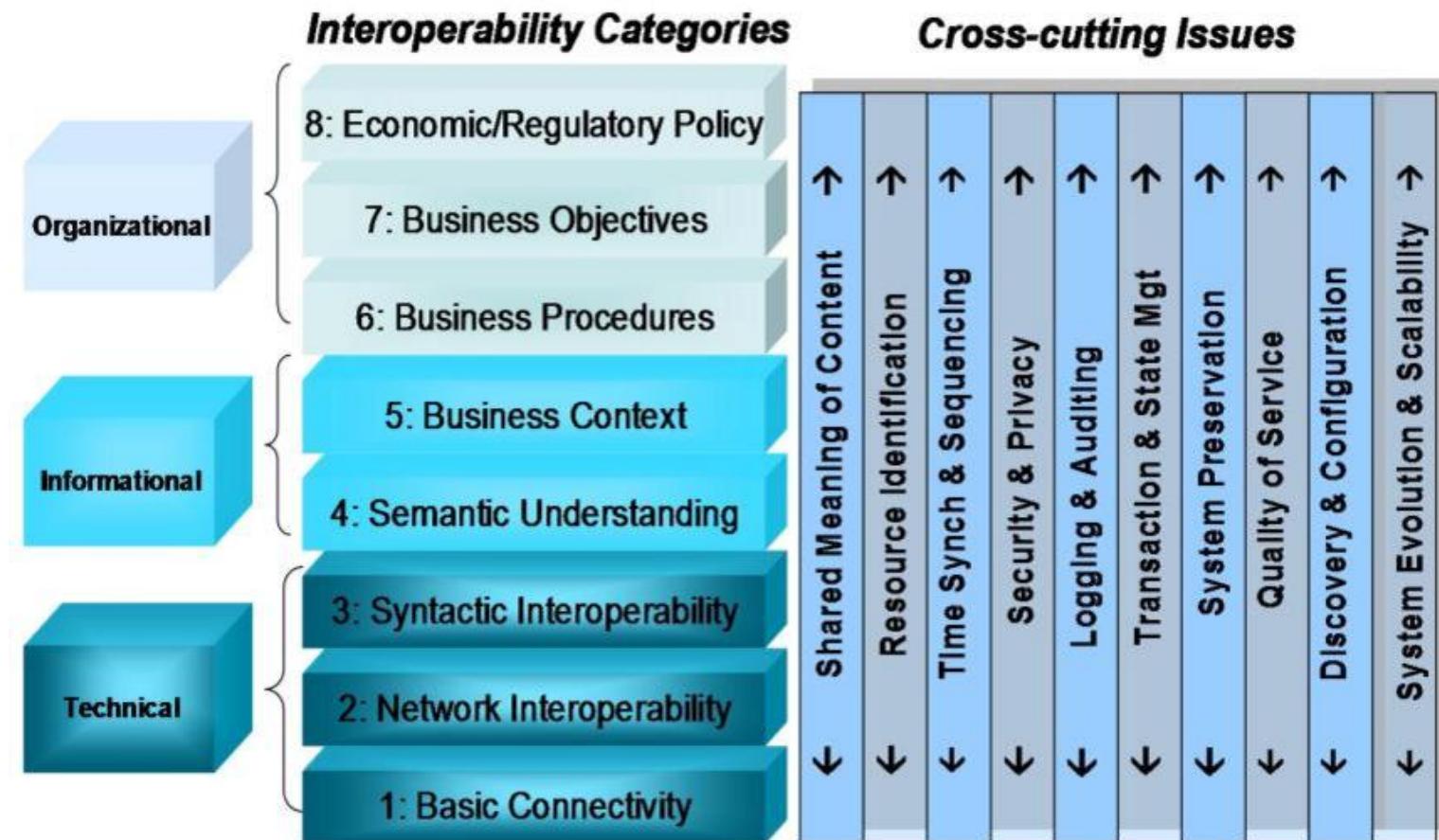
Requisite System Attributes  
and Capabilities



# Enterprise Interoperability Framework



# GridWise Interoperability Context-Setting Framework



- Naudet et al.
- Ova ontologija opisuje sustav ontološkog metamodela, probleme i rješenja i može se koristiti za diagnosticiranje i rješavanje problema interoperabilnosti.
- Dva alternativna tehnička rješenja problema interoperabilnosti: premošćivanje i homogenizacija.
- Premošćivanje koristi međusustav (često zvan adapter) između sustava koji imaju problema s interoperabilnošću.
- Homogenizacija podrazumijeva unificirani model i djeluje izravno na modele ili njihove reprezentacije. Zahtijeva ili sintaktičke ili semantičke transformacije koje koriste definirani jedinstveni model.

- Za sada ne postoje usvojeni standardi računalstva u oblaku među različitim komercijalnim pružateljima usluga u oblaku.
- Svaki pružatelj komercijalnih usluga ima svoje specifične API-je i različita tehnološka rješenja, što ne pogoduje njihovoj međusobnoj interoperabilnosti.
- Stoga su inicijative za standardizaciju u ovom području vrlo važne.
- The Open Cloud Computing Interface (OCCI), The Cloud Infrastructure Management Interface (CIMI), The Open Virtualization Format (OVF), OASIS's Topology and Orchestration Specification for Cloud Applications (TOSCA)

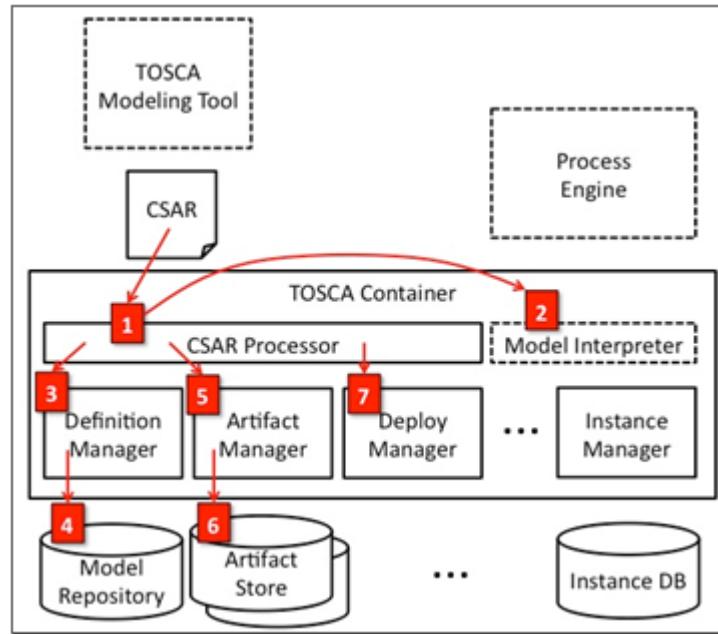
# The Cloud Infrastructure Management Interface (CIMI)



- Definira model za upravljanje resursima infrastrukture kao usluge.
- Bavi se implementacijom i upravljanjem virtualnim strojevima, volumenima, mrežom i drugim IaaS artefaktima.

- DMTF standard koji opisuje otvoreni format za virtualna računala.
- Optimiziran je za distribuciju jednog ili više virtualnih strojeva; neovisan je o dobavljaču i platformi, proširiv je i lokaliziran.
- Pruža potpunu specifikaciju virtualnog stroja.

- TOSCA se može koristiti za pružanje opisa komponenti usluge, njihovih odnosa i postupaka upravljanja uslugama.
- Temeljni koncept TOSCA-e je mogućnost premještanja usluga i aplikacija između javne i privatne infrastrukture oblaka.



- Određuje sučelje za pohranu u oblaku i njihovo uspješno upravljanje
- Programerima u oblaku omogućuje otkrivanje mogućnosti odabrane pohrane u oblaku, upravljanje spremnicima i njima pridruženim podacima te korištenje metapodataka za spremnike i/ili podatkovne objekte.
- CDMI pruža standardizirano sučelje putem RESTful web usluga koje se mogu koristiti za stvaranje, dohvaćanje, ažuriranje i brisanje podatkovnih objekata.
- CDMI je sada prihvaćen kao ISO standard ISO/IEC 17826:2012.

- Ima li smisla standardizacija iznad sloja infrastrukture kao servisa?
- Barijere za standardizaciju oblaka: prepreke za izlaz koje mnogi dobavljači postavljaju u svoje ponude u oblaku, diferencirane usluge raznih komercijalnih dobavljača, standardi sazrijevaju godinama, a za svaki od tri glavna modela računarstva u oblaku potrebni su različiti standardi.

- Semantički web je tehnologija koja se koristi za ostvarivanje semantičke interoperabilnosti
- Koristi se i za semantičku interoperabilnost servisa u oblacima
- Glavna ideja semantičkog weba je osigurati koherentan model podataka koji je dio web infrastrukture
- Temeljni koncepti semantičkog weba su: AAA slogan (svatko može reći bilo što o bilo kojoj temi), otvoreni svijet (podrazumijeva se da uvijek postoji više informacija nego što je poznato) i nejedinstveno imenovanje (istи entitet može imati više imena).

- Semantičko modeliranje obično počinje s definicijom pitanja o kompetencijama kako bi se odredilo na koja bi pitanja model trebao odgovoriti
- Semantički model trebao bi predvidjeti njegovu moguću upotrebu od strane nekoga tko nije njegov dizajner u budućnosti, te bi trebao biti fleksibilan u pogledu mogućnosti nadogradnje i spajanja s drugim semantičkim modelima.
- Glavni jezici semantičkog weba su RDF i OWL

- Resource Description Framework (RDF)
- Osnovni jezik semantičkog weba
- Može se prikazati kao grafikon čvorova i lukova koji prikazuju resurse, njihova svojstva i vrijednosti.
- Ovaj jezik za modeliranje koristi određenu terminologiju za različite dijelove iskaza: subjekt (ono o čemu se govori u iskazu), predikat (svojstvo subjekta) i objekt (vrijednost svojstva)

# Primjer RDF dokumenta

```
<http://example.org/bob#me> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://xmlns.com/foaf/0.1/Person> .

<http://example.org/bob#me> <http://xmlns.com/foaf/0.1/knows>
<http://example.org/alice#me> .

<http://example.org/bob#me> <http://schema.org/birthDate> "1990-07-
"^^<http://www.w3.org/2001/XMLSchema#date> .

<http://example.org/bob#me> <http://xmlns.com/foaf/0.1/topic_interest>
<http://www.wikidata.org/entity/Q12418> .

<http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/title> "Mona
Lisa" .

<http://www.wikidata.org/entity/Q12418> <http://purl.org/dc/terms/creator>
<http://dbpedia.org/resource/Leonardo_da_Vinci> .

<http://data.europeana.eu/item/04802/243FA8618938F4117025F17A8B813C5F9AA4D
619> <http://purl.org/dc/terms/subject> <http://www.wikidata.org/entity/Q12418> .
```

- Web Ontology Language (OWL 2)
- Elementi: klase, svojstva, pojedinci i vrijednosti podataka
- To je semantički jezik dizajniran za predstavljanje bogatog i složenog znanja.
- OWL 2 ontologije mogu se koristiti zajedno s informacijama napisanim u RDF-u.

# Primjer OWL 2

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY PaaSOntologyv5 "http://localhost:8080/PaaSOntologyv5.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>

<rdf:RDF xmlns="http://localhost:8080/PaaSOntologyv4.owl#"
    xml:base="http://localhost:8080/PaaSOntologyv4.owl"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:PaaSOntologyv5="http://localhost:8080/PaaSOntologyv5.owl#">
    <owl:Ontology rdf:about="http://localhost:8080/PaaSOntologyv5.owl#">
        <rdfs:comment>This is ontology of PaaS resources and remote operations.
        Version 5 - adding other cross-PaaS service data types</rdfs:comment>
    </owl:Ontology>
```

- Ontologija je eksplisitna specifikacija konceptualizacije
- Ontologija u OWL-u je skup preciznih izjava o domeni interesa
- Aksiomi u OWL-u su osnovni iskazi OWL ontologije
- Entiteti uključuju sve vrste elemenata koji se koriste za upućivanje na objekte stvarnog svijeta: apstraktne kategorije (klase u OWL-u), relacije (svojstva objekta, svojstva tipa podataka i svojstva napomena u OWL-u) i objekte (pojedince).

- ONIONS (Ontologic Integration of Naive Sources)
- COINS (Context Interchange System)
- METHONTOLOGY
- OTK (On-To-Knowledge)
- Cyc
- TOVE (Toronto Virtual Enterprise)
- IDEF5 (Integrated Definition for Ontology Description Capture Method)
- UPON (United Process for Ontologies)
- Ontology Development 101.

# Ontology Development 101



- Trenutačni web servisi pružaju samo sintaktičke opise, tako da se integracija web servisa mora izvršiti ručno.
- Semantički web servisi su integracija semantičkog weba i servisno orijentirane arhitekture implementirane u obliku web servisa.
- Semantički web servisi usmjereni su na automatizirano rješavanje sljedećih problema: opis, objavljivanje, otkrivanje, posredovanje, praćenje i kompozicija servisa.

- Ontologija servisa koja korisnicima i/ili softverskim agentima omogućuje otkrivanje, pozivanje i kompoziciju web servisa
- Ova ontologija definirana je korištenjem OWL jezika.
- Ima tri glavna dijela: profil servisa za određivanje namjene i funkcionalnosti servisa; model procesa za opisivanje rada servisa i temelj koji sadrži detalje o tome kako koristiti servis.

- WSMO se koristi za opisivanje različitih aspekata povezanih sa servisima semantičkog weba.
- To je proširenje *Web Service Modeling Framework-a* (WSMF).
- Sam WSMF sastoji se od četiri različita elementa: ontologije, ciljeva, opisa web servisa i posrednika.
- WSMO dorađuje i proširuje ovaj okvir razvijanjem ontologije za ključne elemente servisa semantičkog weba i jezika opisa koji se sastoji od nefunkcionalnih, funkcionalnih i bihevioralnih aspekata web servisa.

# „Lagane“ servisne ontologije (engl. Lightweight service ontologies )

- Brži rad na semantičkom anotiranju
- WSMO-Lite
- SAWSDL
- MicroWSMO
- hRESTS
- SA-REST

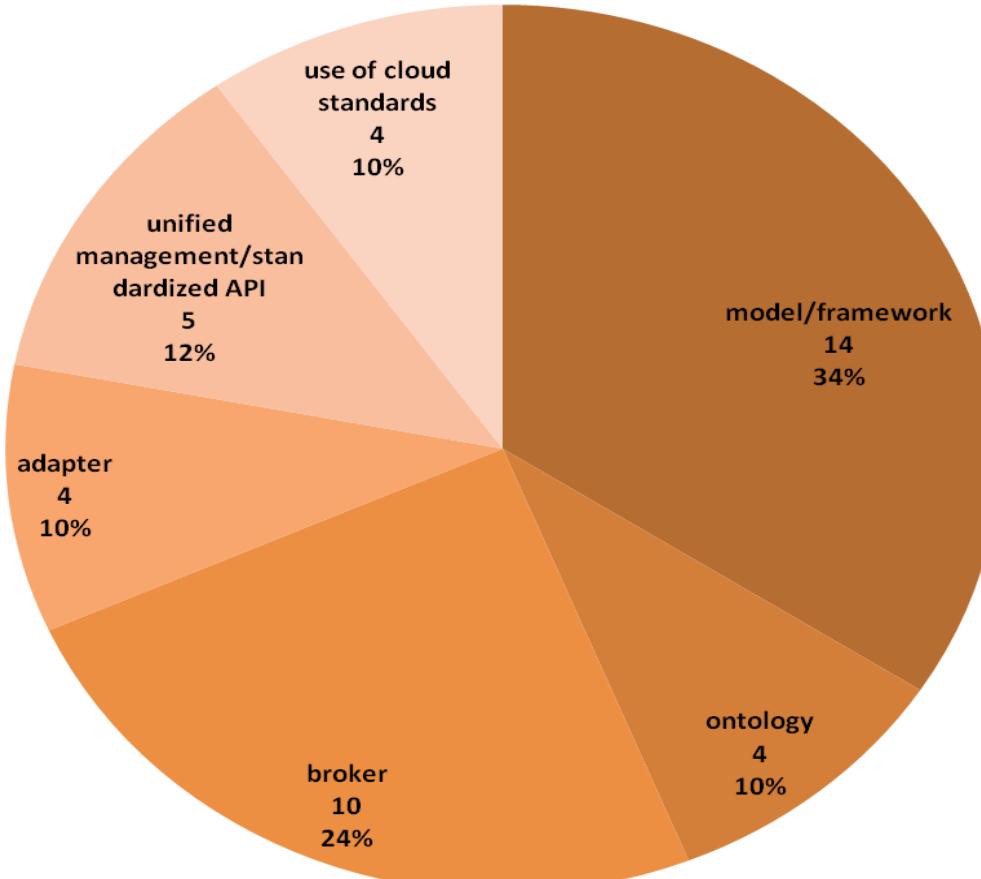
- Event Calculus
- Petri Nets
- Colored Petri Nets
- Linear Logic theorem proving
- AI planning
- logic programming
- Markov process
- States Machines

- Hierarchical Task Network (HTN planiranje)
- HTN planer koristi znanje o domeni i formulira plan rekurzivnom dekompozicijom zadatka dok ne dođe do primitivnih zadatka koji se mogu izravno izvršiti
- SHOP2 planer
- JSHOP
- GTPyhop - Goal-Task-Network sustav za planiranje – nasljednik SHOP2-a

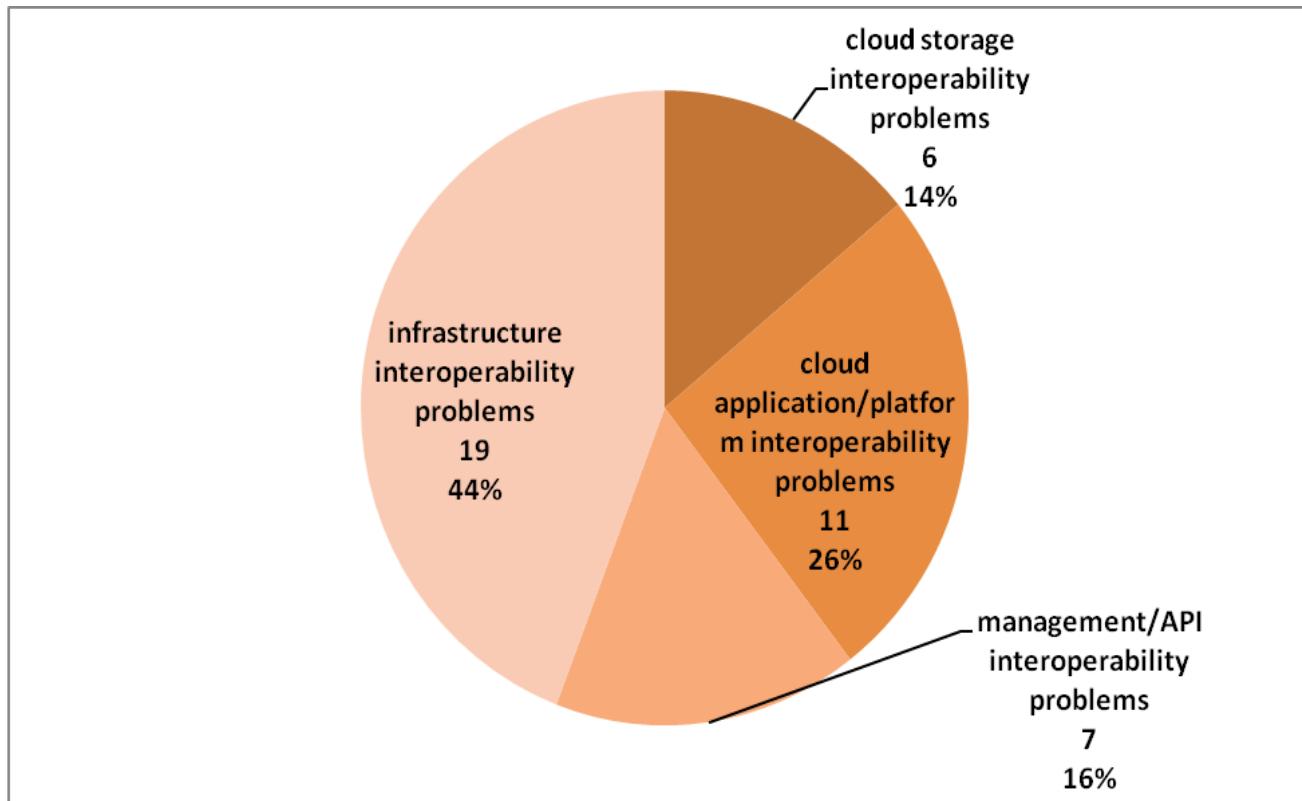
- Različite definicije interoperabilnosti oblaka: sposobnosti modeliranja programskih razlika, prevodenja između različitih apstrakcija oblaka, premještanja aplikacija iz jednog oblaka u drugi, omogućavanja aplikacijama da rade na više oblaka, prijenosa podataka između pružatelja usluga oblaka, korištenja objedinjenih alata za upravljanje za višestruke oblake

- Cloud4SOA
- mOSAIC
- Contrail
- Vision Cloud
- REMICS
- MODAClouds

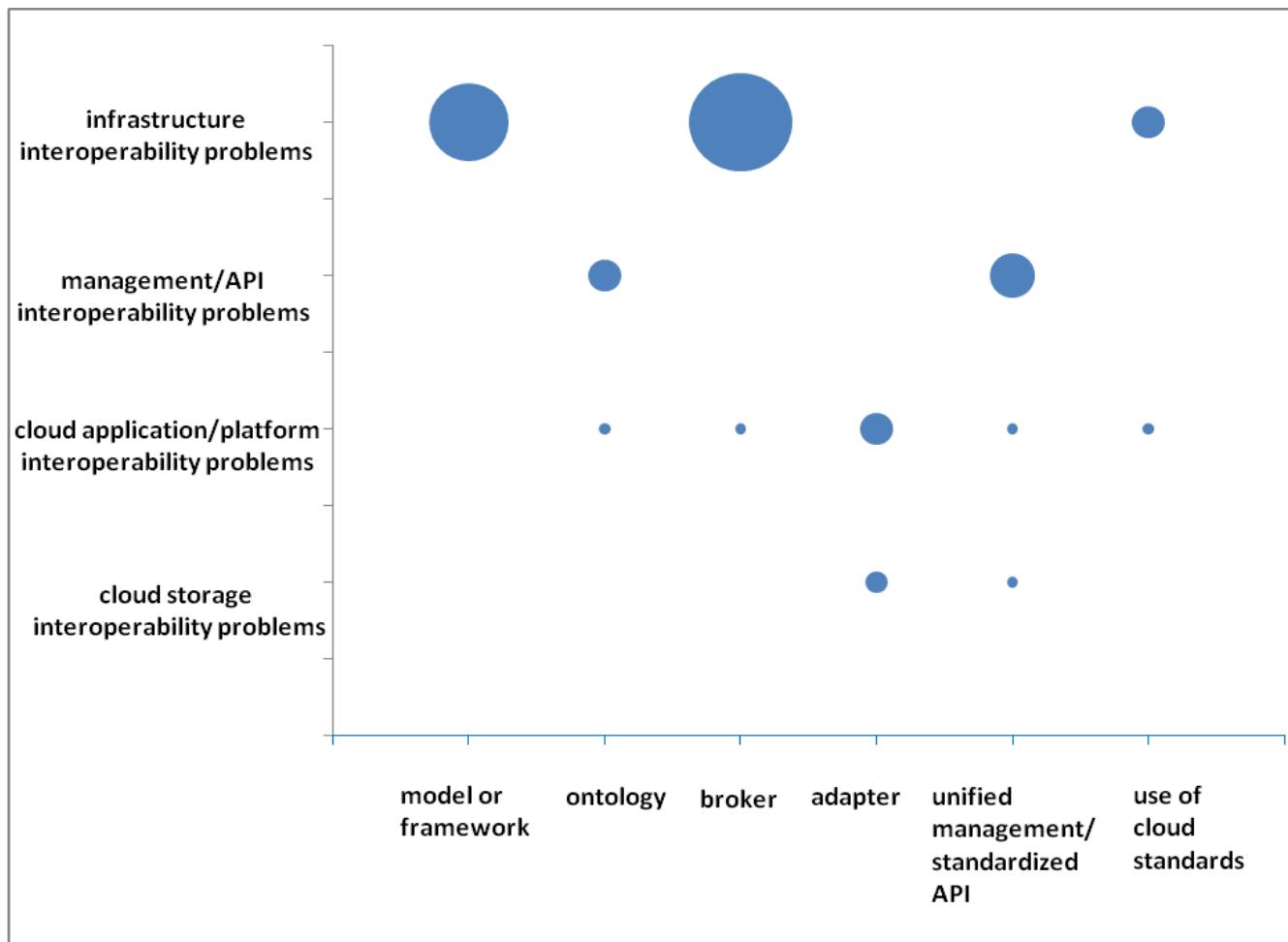
# Pregled istraživanja o interoperabilnosti oblaka (1)



# Pregled istraživanja o interoperabilnosti oblaka (2)



# Pregled istraživanja o interoperabilnosti oblaka (3)



# Pitanja i rasprava



# Interoperabilnost servisa interneta stvari i servisa na cloudu

Darko Andročec

# Prvi slučaj korištenja: Migracija podataka između oblaka



ID slučaja korištenja: UC-1

Naziv slučaja korištenja: Migracija podataka između različitih pružatelja usluga u oblacima

Sudionici: Cloud korisnik, pružatelj usluga 1, pružatelj usluga 2

Opis: Ovaj slučaj upotrebe pokazuje kako premjestiti podatke s jednog cloud pružatelja na drugog. Korisnik može odlučiti premjestiti sve podatke ili samo jedan spremnik podataka (tablica, entitet ili prilagođeni objekt) od izvornog cloud dobavljača do ciljanog cloud dobavljača.

Okidač: Ovaj slučaj upotrebe pokreće korisnik oblaka kada odluči premjestiti podatke pohranjene u trenutnoj cloud ponudi u drugu.

Preduvjeti: 1. Cloud korisnik mora imati postojeće podatke pohranjene na jednoj cloud ponudi  
2. Cloud korisnik mora se registrirati na drugu cloud ponudu i moći unositi podatke na nju

Postuvjeti: 1. Odabrani podaci premještaju se iz jedne cloud ponude u drugu

Normalni tijek: 1. Cloud korisnik odabire želi li premjestiti sve podatke ili određeni spremnik podataka (tablicu, entitet ili prilagođeni objekt)  
2. Cloud korisnik odabire izvornu i ciljnu Cloud ponudu među dostupnim  
3. Cloud korisnik pokreće migraciju podataka

Alternativni tijekovi: -

Iznimke: 1. Ako postoji problem s povezivanjem s odabranim izvornim ili ciljnim cloud ponudama, pokreće se iznimka i prikazuje se poruka o pogrešci  
2. Ako sustav pronađe problem interoperabilnosti tijekom migracije podataka, migracija podataka se zaustavlja i pronađeni problem interoperabilnosti prikazuje se cloud korisniku

Uključuje: Nijedan drugi slučaj upotrebe nije uključen u ovaj slučaj upotrebe.

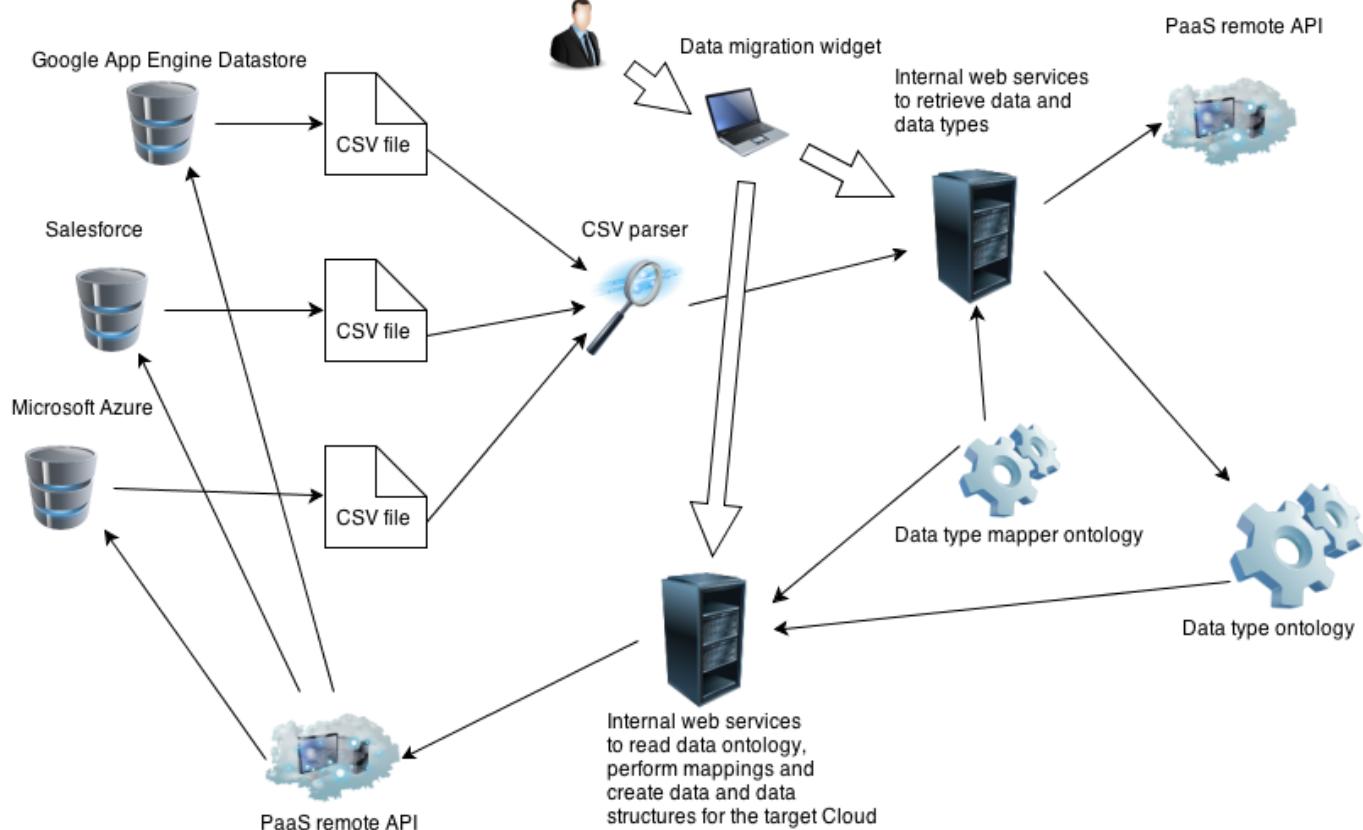
Specijalni zahtjevi: Migracija podataka trebala bi biti fleksibilna i koristiti cloud ontologiju i preslikavanja tipova podataka koji su u njima definirani.

Prepostavke: Cloud korisnik razumije engleski jezik.

Bilješke i problemi: -

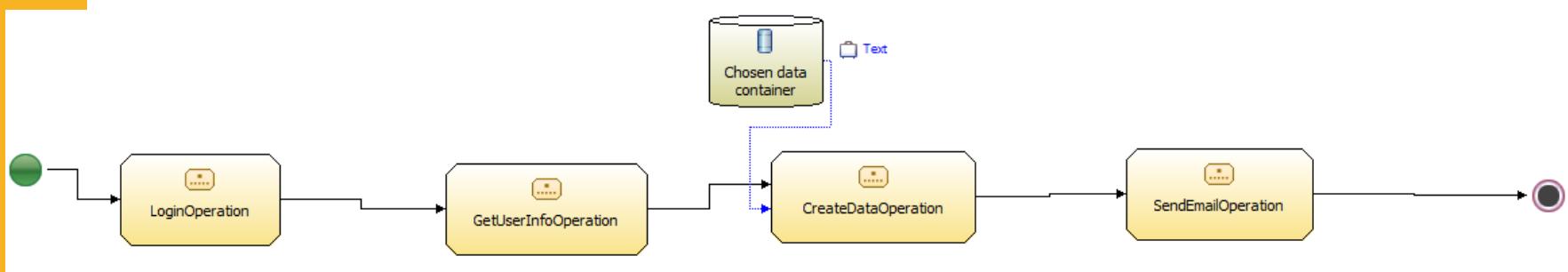
- Izvoz podatkovnih struktura i podataka od pružatelja usluge u oblaku (npr. izvoz CSV-a)
- Transformacija struktura podataka i podataka u ontologiju
- Mapiranje tipova podataka – jezik posrednik OWL

# Primjer arhitekture migracije podataka



- Razlike između modela pohrane podataka različitih pružatelja usluga u oblaku – npr. teško je ili gotovo nemoguće migrirati podatke iz SQL modela u NoSQL model
- Nestandardizirani API-ji
- Neki pružatelji koriste svoju vlastiti jezik za upite – npr. Salesforce Object Query Language (SOQL), Salesforce Object Search Language (SOSL), Google Query Language (GQL)

# Drugi slučaj korištenja: Dodavanje informacija o korisniku iz jednog u drugi cloud



# Opis slučaja korištenja 2



Naziv slučaja korištenja: Dodavanje postojećeg korisnika u drugi oblak

Sudionici: Administrator PaaS aplikacije, PaaS pružatelj usluge 1, PaaS pružatelj usluge 2

Opis: Ovaj slučaj upotrebe pokazuje kako dodati korisnika iz jedne PaaS ponude u aplikaciju postavljenu na drugom PaaS-u. Administrator PaaS-a određuje spremnik podataka ciljanog PaaS-a gdje će se pohraniti podaci o korisniku. Treba također izraditi odgovarajuće datoteke za mapiranje sheme. Na kraju se administratoru PaaS aplikacije šalje e-mail da je dodan novi korisnik.

Okidač: Ovaj slučaj upotrebe pokreće administrator PaaS aplikacije kada odluči da želi dodati postojeće korisničke podatke (iz druge PaaS ponude) PaaS aplikaciji kojom upravlja.

Preduvjeti:

1. Korisnik od kojeg se traži migracija mora biti prijavljen na izvornoj PaaS ponudi
2. Administrator PaaS aplikacije mora moći staviti podatke u spremnik podataka s korisničkim informacijama ciljne PaaS ponude

Postuvjeti:

1. Postojeći korisnik iz izvorne PaaS ponude dodaje se aplikaciji postavljenoj na ciljnoj PaaS ponudi, e-pošta se šalje administratoru PaaS aplikacije1.

Normalni tijek:

1. Administrator PaaS aplikacije odabire izvor i veze ciljane PaaS ponude, te navodi naziv spremnika podataka gdje se pohranjuju korisničke informacije za ciljnu aplikaciju
2. Administrator PaaS aplikacije pokreće migraciju korisnika
3. Izvode se ulazno/izlazna mapiranja, pozivaju se odgovarajuće web usluge, dodaje se korisnik za aplikaciju pohranjenu u ciljnoj PaaS ponudi, a e-pošta o novom korisniku šalje se administratoru

Alternativni tijekovi:

Iznimke:

1. Ako postoji problem s povezivanjem s odabranim izvornim ili ciljnim PaaS ponudama, pokreće se iznimka i prikazuje se poruka o pogrešci
2. Ako sustav pronađe problem interoperabilnosti tijekom faze planiranja ili izvršenja servisa, radnja se zaustavlja i pronađeni problemi interoperabilnosti prikazuju se u korisničkom sučelju

Uključuje:

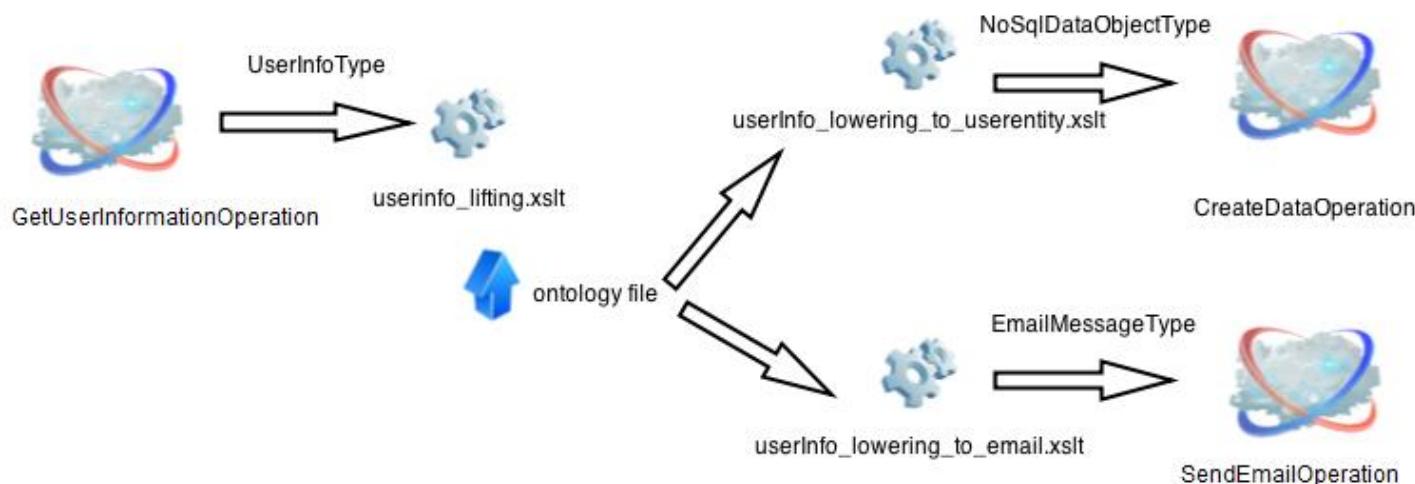
Nijedan drugi slučaj upotrebe nije uključen u ovom slučaju.

Specijalni zahtjevi:

Ovaj slučaj upotrebe trebao bi potvrditi interoperabilnost razine usluge API-ja, koristeći posredovanje podataka temeljeno na ontologiji i shemu podizanja i spuštanja kako je definirano u SAWSDL-u.

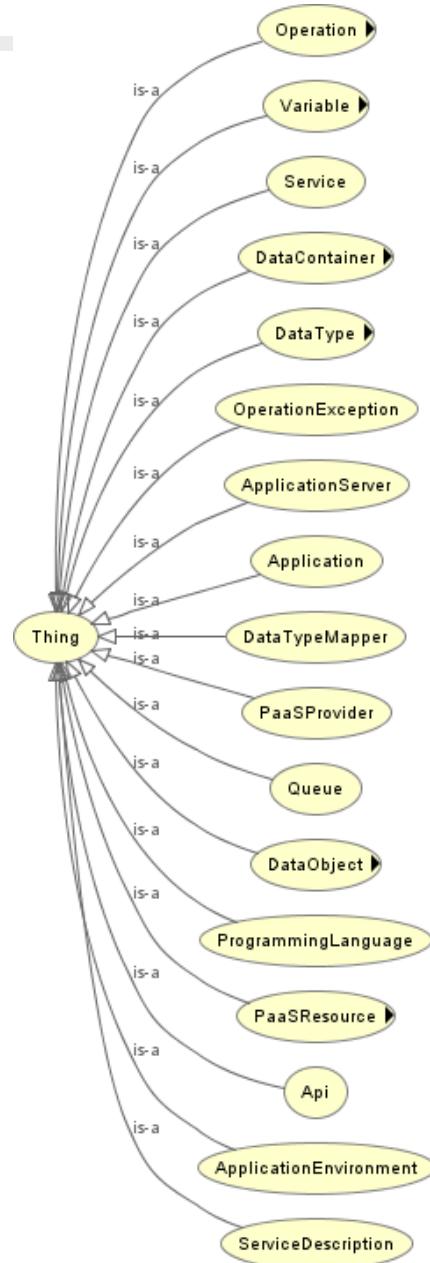
# Postupak implementacije 2. slučaja korištenja

- Podatkovno posredovanje putem ontologije

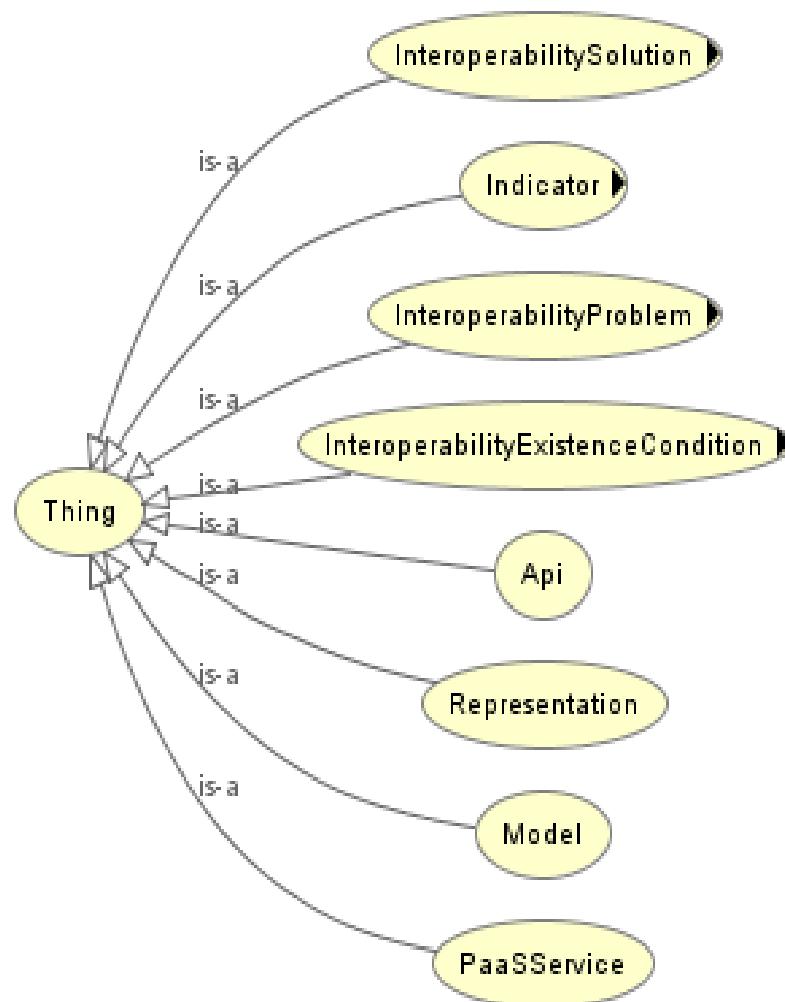


- API operacije različitih PaaS dobavljača imaju različite tipove, najčešće su ti tipovi složeni (sastoje se od više jednostavnih tipova i/ili drugih složenih tipova).
- Kako bi se postigla interoperabilnost, potrebno je definirati preslikavanja i transformacije između ulaza i izlaza.
- Korištenje među-PaaS koncepta za tipove podataka u ontologiji pojednostavljuje preslikavanja, te omogućuje kreiranje novih preslikavanja i mogućih transformacija, kada se koristi nova PaaS ponuda ili kada se mijenja određeni API.
- Ovo je fleksibilniji pristup od pristupa izravnog mapiranja i transformacije koji se koristi u jezicima za kompoziciju web servisa kao što je BPEL.
- Najkritičniji dio ovog pristupa je zahtjev da korisnik/administrator stvori valjana i smislena mapiranja i transformacije.

# Ontologija PaaS resursa i operacija



# Ontologija problema interoperabilnosti



# Predložena metodika

Step	Activities
<b>1. Requirement identification</b>	1.1 Choose cloud model 1.2 Study the existing use cases 1.3 Identification of relevant interoperability actions 1.4 Define use cases
<b>2. Interoperability analysis</b>	2.1 Review the existing literature on interoperability problems 2.2 Identify levels of interoperability problems 2.3 Identify specific interoperability issues at each level 2.4 Choose ontology development methodology 2.5 Create ontology of interoperability problems
<b>3. Solution design</b>	3.1 Create ontology of resources, remote operations, and data types 3.2 Choose language for semantic web services 3.3 Create mappings and transformations 3.4 Associate mappings and transformations to the appropriate elements of services 3.5 Choose AI planner 3.6 Define AI planning domain 3.7 Define algorithms for finding interoperability problems
<b>4. Solution implementation</b>	Use cases execution: 4.1 Implement needed web services to invoke remote APIs 4.2 Generate AI planning problem based on semantic annotations, the ontology and user choice 4.3 Develop or modify/upgrade interoperability tool 4.4 Get a suitable plan from AI planner or find interoperability problems 4.5 Execute service composition
<b>5. Evaluation</b>	5.1 Evaluation of the ontologies 5.2 Validation of execution of use cases

- Using JSON-LD to Compose Different IoT and Cloud Services – Darko Andročec -  
<https://arxiv.org/abs/1809.08233>
- Kompozicija različitih stvari
- Primjer: jednostavna aplikacija koja će LED svjetlom vizualno pokazati kada je temperatura u prostoriji preniska ili previsoka

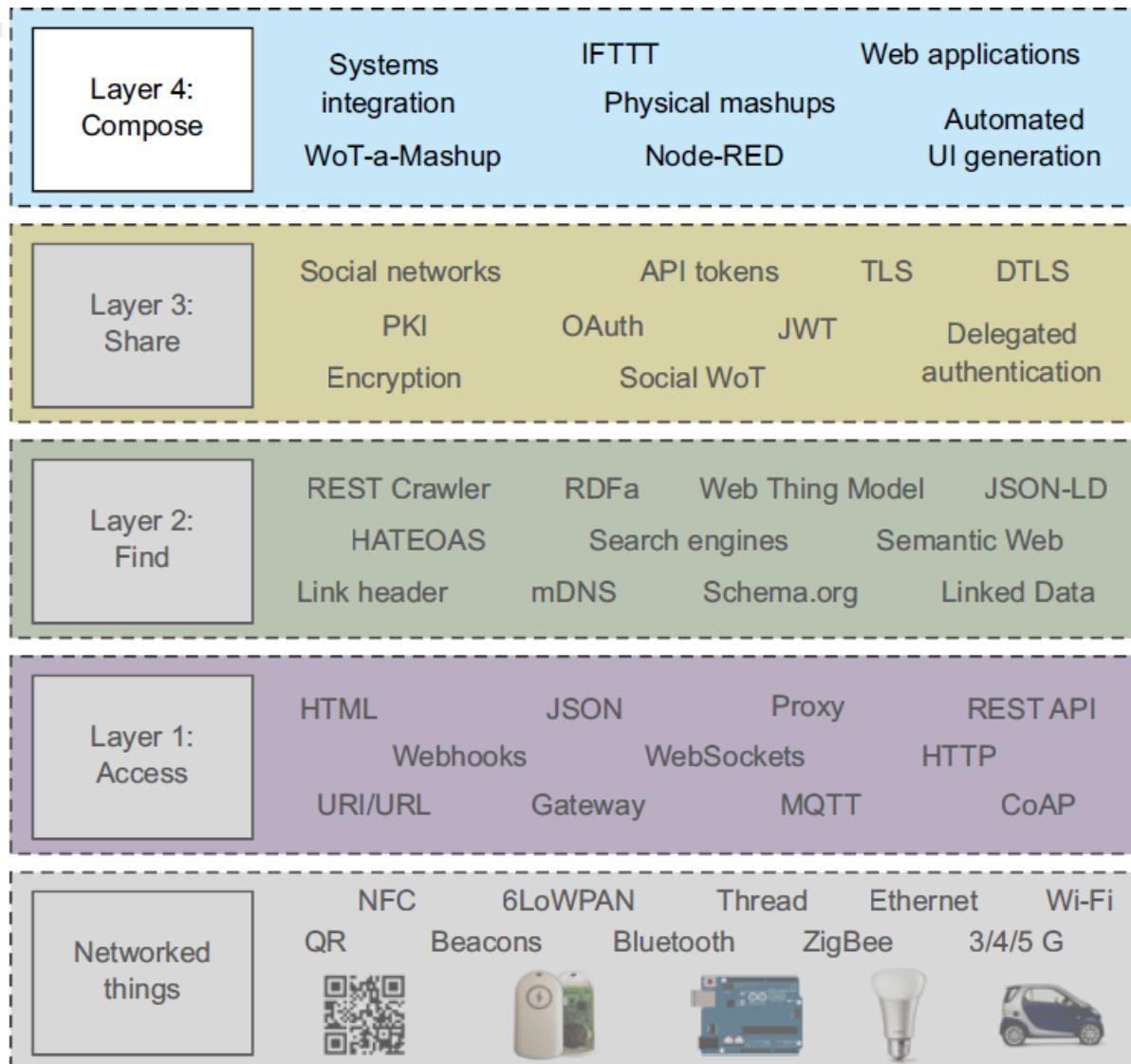
- Na temelju akcije koju je odabrao korisnik (popis zadataka koji će se izvršiti), SAWSDL, JSON-LD datoteke i IoT ontologija analiziraju se za generiranje logičkih atoma.
- SAWSDL parser je razvijen u Javi korištenjem EasyWSDL biblioteke otvorenog koda i njenog proširenja EasySAWSVL.
- Također, razvijena je Java klasa JSON-LD parsera koja koristi JSON i JSON-LD biblioteke.
- Klasa za parsiranje OWL ontologije implementirana je korištenjem Apache Jena biblioteke. Datoteka opisa domene sastoji se od operatora, metoda i aksioma.
- Datoteka opisa domene se definira ručno.

- (defdomain iot (
- ; BEGIN compose temperature sensor and LED actuator
- (:method (composeIoTServices ?sensor ?actuator)
- ()
- (!checkSensorActuator ?sensor ?actuator))
- )
- (:operator (!checkSensorActuator ?sensor ?actuator)
- ((SemanticWebThing ?thingId1) (hasResources ?thingId1 Sensor ?sensor) (SemanticWebThing ?thingId2) (hasResources ?thingId2 Actuator ?actuator) )
- ()
- (sensorCanConnectToActuator ?sensor ?actuator))
- )
- ; END compose temperature sensor and LED actuator
- )
- )

# Primjer isječka JSON-LD datoteke

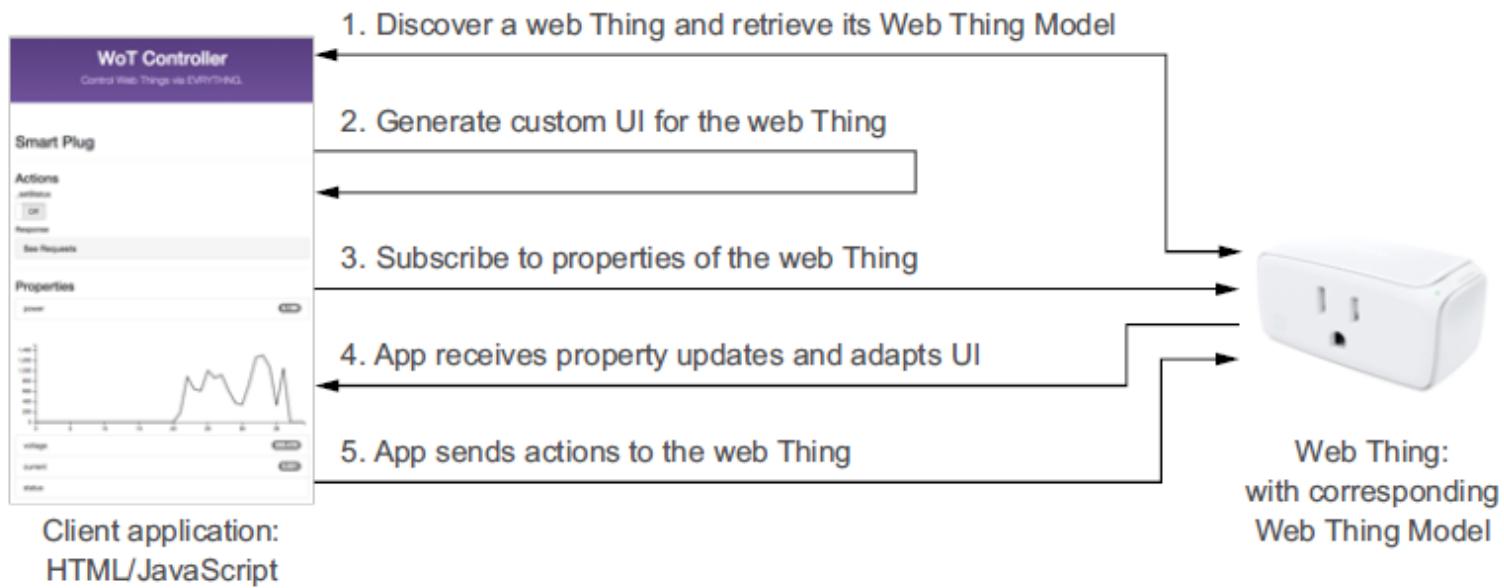
```
"@context":  
{  
  "rdf": "http://www.w3.org/1999/02/22-rdf-syntax-ns#",  
  "rdfs": "http://www.w3.org/2000/01/rdf-schema#",  
  "owl": "http://www.w3.org/2002/07/owl#",  
  "myont": "http://iot.foi.hr/ontologies/ThingAsAServiceOntology.owl#"  
},  
"@type": "myont.SemanticWebThing",  
"thingId": {"value": "2341"},  
"thingName": {"value": "littleBits cloud bit"},  
"thingDescription": {"value": "littleBits with LED"},  
"myont.hasResources": {  
  "@type": "myont.Actuator",  
  "poName": "02 long LED",  
  "poDescription": "long white LED",  
  "hasValues" : {  
    "@type": "Input",  
    "inputName" : "input voltage",  
    "inputDescription": "input voltage in percentage",  
    "inputUnit": "percent"  
  }  
},  
"myont.supportsProtocols": {  
  "@type": "myont.IoTProtocols",  
  "proName": "WiFi",  
  "proDescription" : "WiFi"  
},  
}
```

# Kompozicija fizičkih stvari



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi* 17

# Jedinstveno sučelje za WoT



Izvor: *Building the Web of Things: With examples in Node.js and Raspberry Pi*

- Web mashup je aplikacija koja zahtijeva nekoliko web resursa i koristi ih za izradu nove, hibridne aplikacije
- Koncept mashupa također može primijeniti na Web of Things, u obliku fizičkih kombinacija (engl. *physical mashup*)
- To su web aplikacije koje kombiniraju servise iz fizičkih stvari sa servisima iz virtualnih web izvora.

- Node – RED
- <https://nodered.org/>
- Programiranje niskog koda (engl. Low-code programming) za aplikacije vođene događajima
- Node-RED je alat za programiranje za spajanje hardverskih uređaja, API-ja i online servisa na nove i zanimljive načine.
- Omogućuje uređivač temeljen na pregledniku koji olakšava spajanje tokova pomoću širokog raspona čvorova u paleti koji se mogu postaviti u vrijeme izvođenja jednim klikom.
- <https://youtu.be/ksGeUD26Mw0>

- Mashup alati temeljeni na čarobnjaku sastoje se od korisničkog sučelja koje vas vodi kroz niz koraka za stvaranje prilagođenog tijeka rada.
- IFTTT - “If This Then That”
- if (conditions) then (actions)
- <https://ifttt.com/>
- IFTTT Maker kanal – mogućnost poziva REST servisa treće strane, uključujući i WoT API-je



# Sigurnosni rizici i problemi weba stvari

Darko Andročec

- Trenutno ima na milijarde IoT uređaja
- Često se na sigurnost takvih uređaja misli sasvim na kraju
- Često su sustavi toliko ograničeni da je izgradnja sigurnosti na razini organizacije kakva je implementirana na modernim web i PC sustavima teška ako ne i nemoguća na jednostavnim IoT senzorima.
- Sigurnost na razini: fizičkih uređaja, komunikacijskih sustava i mreža

# Primjeri IoT napada

- Mirai
- Stuxnet
- Chain Reaction

- Najštetniji napad uskraćivanjem usluge (DDoS) u povijesti koji je proizašao iz nesigurnih IoT uređaja u udaljenim područjima.
- Mirai - naziv malicioznog softvera – 2016
- Ukupno je 600.000 IoT uređaja zaraženo u sklopu botnet-a

- Državnikibernetičko oružje koje cilja industrijske SCADA IoT uređaje koji kontroliraju i rade značajnu i nepovratnu štetu iranskom nuklearnom programu.
- Crv koji oštećuje SCADA Siemensove PLC (engl. Programmable Logic Controllers) i koristi rootkit da mijenja rotacijsku brzinu motora kojeg kontrolira PLC
- Ciljani su bili PLC-ovi koji se koriste za pumpe i centrifuge plinova za obogaćivanje urana
- Napad je bio 2010.

- Chain Reaction je akademska studija koja pokazuje novu vrstu kibernetičkih napada usmjerenih na PAN mesh mreže koji se mogu izvršiti bez ikakve veze s internetom.
- Osim toga, pokazuje koliko ranjivi mogu biti daljinski IoT senzori i kontrolni sustavi.
- Vektor napada bile su žarulje Philips Hue koje se obično nalaze u domovima potrošača kojima se može upravljati putem interneta i aplikacija za pametne telefone.
- Svjetla Philips Hue koriste Zigbee protokol

- Mnoge implementacije IoT-a bit će u udaljenim i izoliranim područjima ostavljajući senzore i rubne usmjerivače ranjivima na fizički napad.
- Osim toga, sam hardver treba moderne zaštitne mehanizme uobičajene u procesorima i strujnim krugovima mobilnih uređaja i osobne elektronike.

- Javni i privatni ključevi ključni su za osiguravanje sigurnog sustava.
- Sami ključevi trebaju odgovarajuće upravljanje kako bi se osigurala njihova sigurnost.
- Postoje hardverski standardi za sigurnost ključeva, a jedan posebno popularan mehanizam je *Trusted Platform Module* (TPM).
- TPM je diskretna hardverska komponenta s tajnim RSA ključem ugrađenim u uređaj tijekom proizvodnje.

## Primjeri napada (1)

- Timing Attacks - Pokušaji iskorištavanja malih razlika u vremenskom rasporedu algoritama. Na primjer, mjerjenje vremena algoritma za dekodiranje lozinke i promatranje ranih izlazaka iz rutine. Napadači također mogu promatrati korištenje predmemorije kako bi svjedočili karakteristikama algoritma.
- Simple Power Analysis (SPA) - Slično vremenskom napadu, ali mjeri velike promjene u dinamičkoj snazi ili struji zbog ponašanja algoritma i operativnih kodova. Javni ključevi su posebno osjetljivi. Analizi je potrebno nekoliko tragova da bi radila, ali tragi trebaju visok stupanj preciznosti. Budući da je većina kriptografskih algoritama matematički zahtjevna, različiti operativni kodovi će se prikazati kao različiti potpisi snage u tragu.<sup>9</sup>

## Primjeri napada (2)

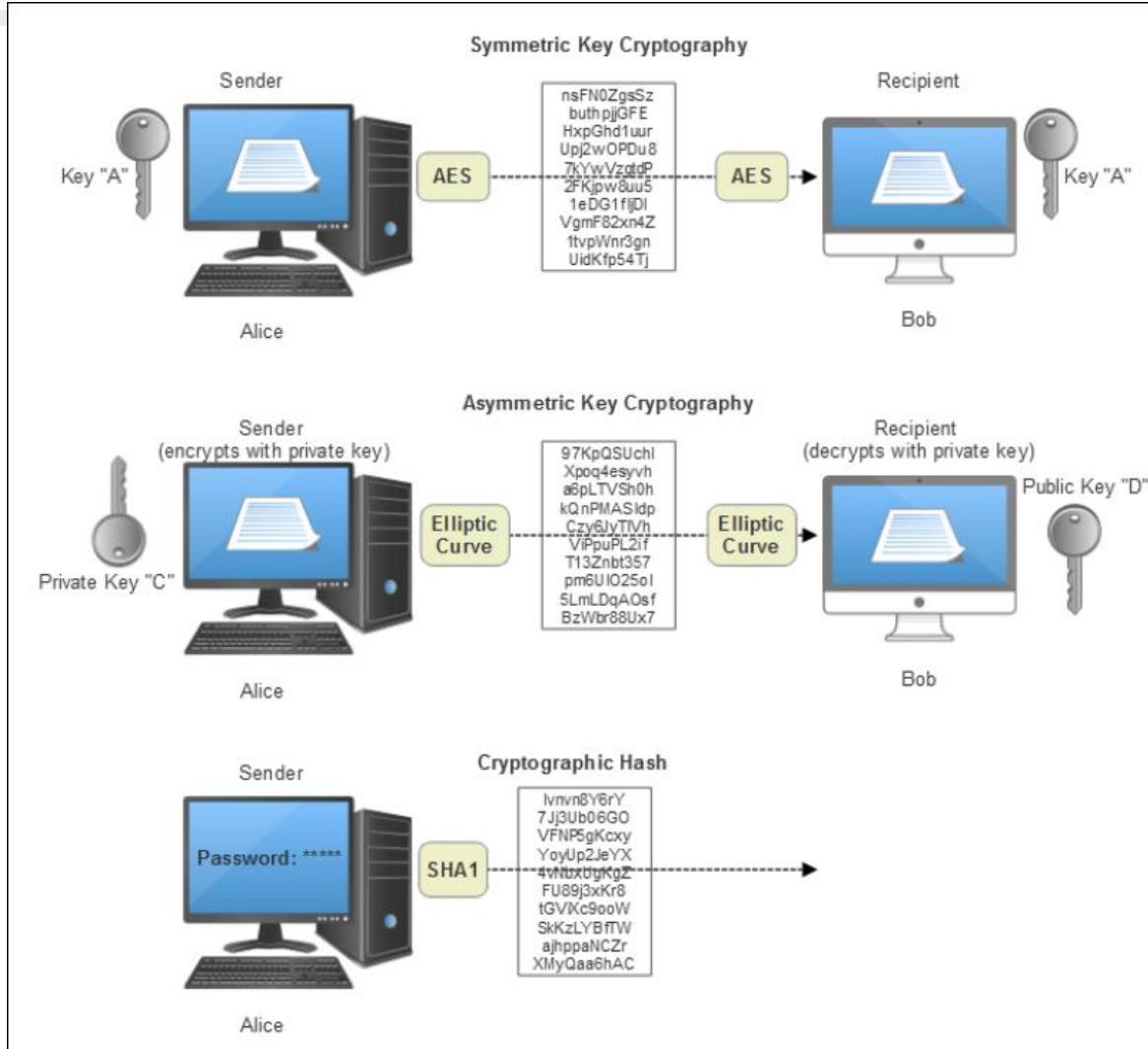
- Differential Power Analysis (DPA) - DPA mjeri dinamičku snagu, ali može promatrati i promjene koje su pre male za SPA. Ubacivanjem nasumičnog unosa (kao što su različiti nasumični ključevi) u sustav, napadač može izvesti tisuće tragova kako bi izgradio skup ovisan o podacima. Napad na AES algoritam, na primjer, jednostavno znači izgradnju dva skupa tragova ovisno o vrijednosti bita (0 ili 1) koji se crackira. Izračunava se prosjek skupova, a razlika između skupa 0 i 1 iscrtana je kako bi se pokazao učinak slučajnog unosa na izlaz.

## Protumjere za ove napade:

- Upotrijebite različite nasumične operacijske kodove kako biste stvorili veliku radnu funkciju za napade.
- Uklonite uvjetne grane koje ovise o ključu.
- Ograničite broj operacija po ključu.
- Koristite operacije promjenjivog vremena ili nagnute satove (engl. *skew clocks*).
- Promjena redoslijeda nezavisnih operacija.

- Enkripcija i tajnost apsolutni su zahtjevi za implementaciju IoT-a.
- Koriste se za osiguranje komunikacije, zaštitu firmvera i autentifikaciju.
- Simetrično šifriranje ključem: ključevi za šifriranje i dešifriranje su identični. RC5, DES, 3DES i AES su svi oblici šifriranja simetričnog ključa.
- Enkripcija s javnim ključem – asimetrična enkripcija: Ključ za šifriranje je javno objavljen kako bi svatko mogao koristiti i šifrirati podatke. Samo primatelj ima privatni ključ koji se koristi za dešifriranje poruke - Elliptic Curve, PGP, RSA, TLS, S/MIME
- Kriptografski hash: Preslikava podatke proizvoljne veličine u bitni niz (koji se naziva sažetak). Ova hash funkcija osmišljena je kao "jednosmjerna," - MD5, SHA1, SHA2, SHA3<sup>12</sup>

# Vrste kriptografije



- Današnji standard za simetričnu enkripciju
- Zamijenio je starije DES algoritme
- Dio je FIPS specifikacije i ISO/IEC 18033-3 standarda

# Pseudokod za AES

```
// Pseudo code for an AES-128 Cipher
// in: 128 bits (plaintext)
// out: 128 bits (ciphertext)
// w: 44 words, 32 bits each (expanded key)
state = in
w=KeyExpansion(key) //Key Expansion phase (effectively encrypts key itself)
AddRoundKey(state, w[0, Nb-1]) //Initial Round
for round = 1 step 1 to Nr-1 //128 bit= 10 rounds, 192 bit = 12 rounds, 256
bit = 14 rounds
SubBytes(state) //Provide non-linearity in the cipher
ShiftRows(state) //Avoids columns being encrypted independently which can
weaken the algorithm
MixColumns(state) //Transforms each column and adds diffusion to the
cipher
AddRoundKey(state, w[round*Nb, (round+1)*Nb-1]) //Generates a subkey an
combines it with state.
end for
SubBytes(state) //Final round and cleanup.
ShiftRows(state)
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
out = state
```

- Naziva se i kriptografijom javnog ključa
- Ključevi mogu biti međusobno zamjenjivi, što znači da ključ može i šifrirati i dešifrirati, ali to nije uvjet.
- Međutim, tipična upotreba je generiranje para ključeva i čuvanje jednog kao privatnog, a drugog kao javnog.
- Tri temeljna kriptograma s javnim ključem: RSA (Rivest-Shamir-Adleman), Diffie-Hellman i Elliptical Curves.

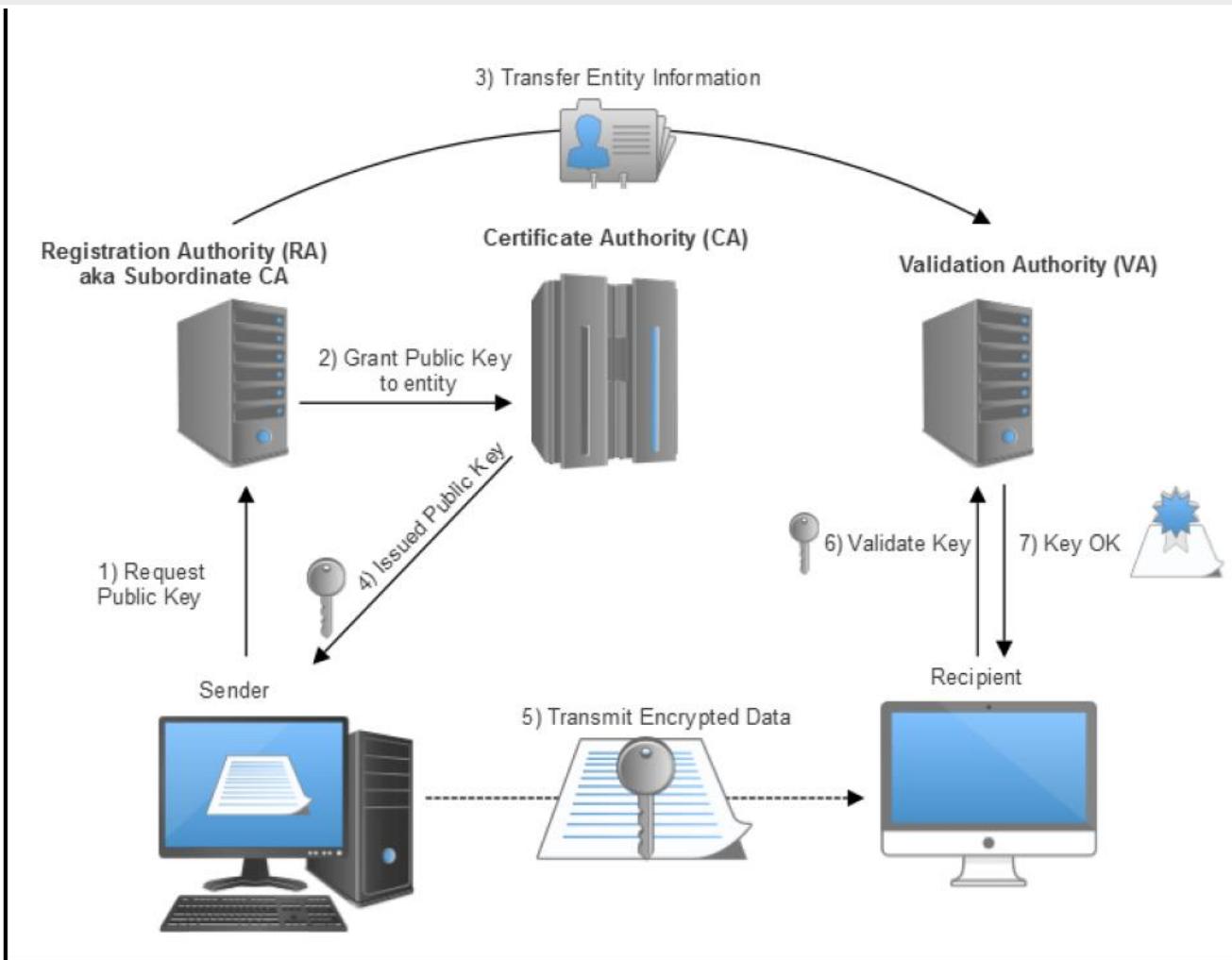
# Kriptografski hash

- Uglavnom se koriste za digitalne potpise
- Također se smatraju "jednosmjernim" ili ih je nemoguće preokrenuti.
- Ponovno stvaranje izvornih podataka nakon prolaska kroz *hash* funkciju bio bi brutalni napad (engl. *brute force*) svake moguće kombinacije ulaza.
- Mala promjena ulaza rezultirat će značajnom entropijom ili promjenom izlaza.
- Dvije različite poruke nikada neće imati istu *hash* vrijednost

# Infrastruktura javnog ključa (1)

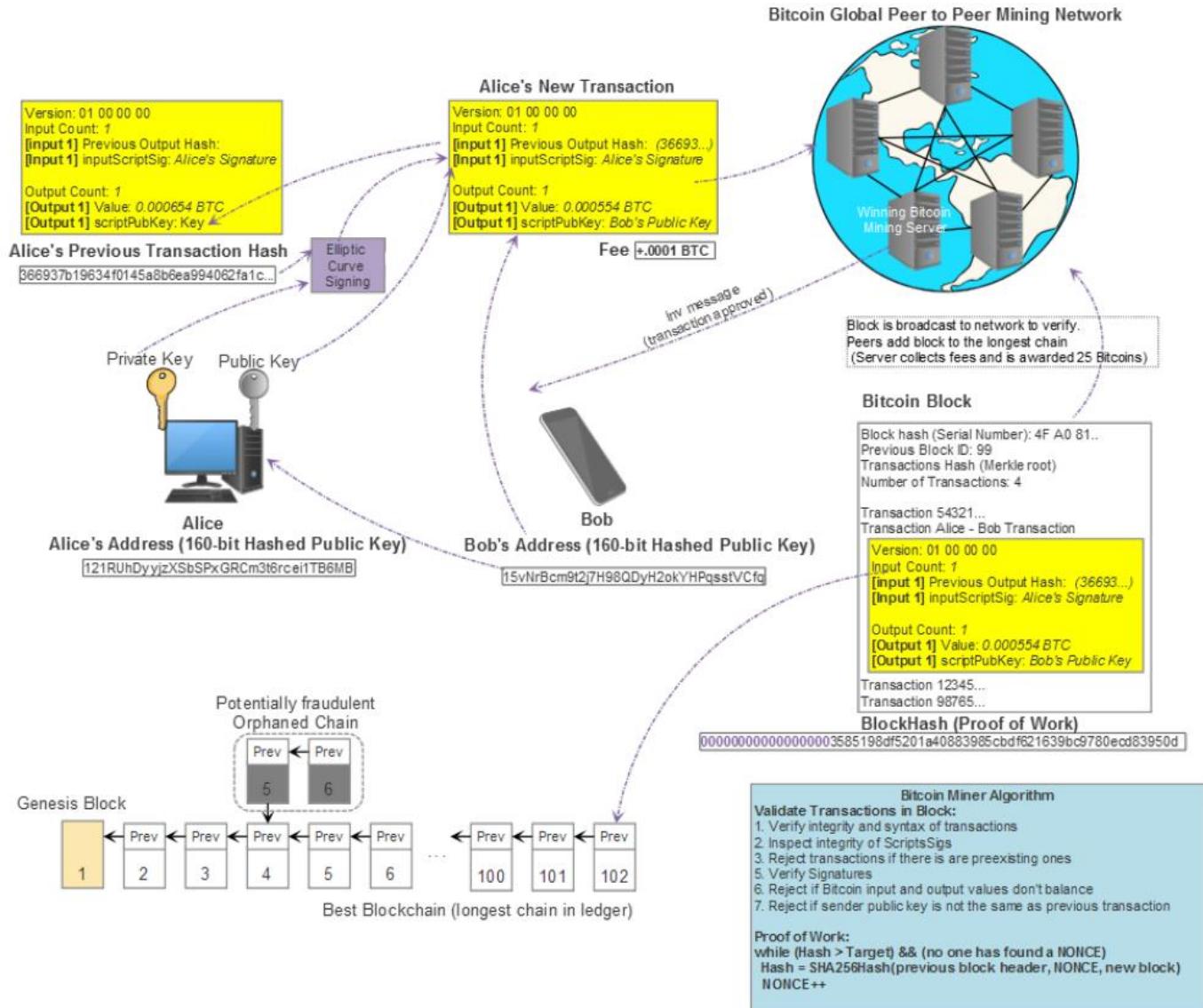
- Asimetrična kriptografija (javni ključ) glavno je uporište internetske trgovine i komunikacije.
- Rutinski se koristi za SSL i TLS vezu na webu.
- Tipična uporaba je šifriranje javnim ključem, gdje podatke u prijenosu šifrira svatko tko ima javni ključ, ali ih može dešifrirati samo vlasnik privatnog ključa.
- Druga upotreba su digitalni potpisi, gdje je BLOB podataka potpisani privatnim ključem pošiljatelja, a primatelj može potvrditi autentičnost ako drži javni ključ.

# Infrastruktura javnog ključa (2)



Iz knjige: Internet of Things for Architects

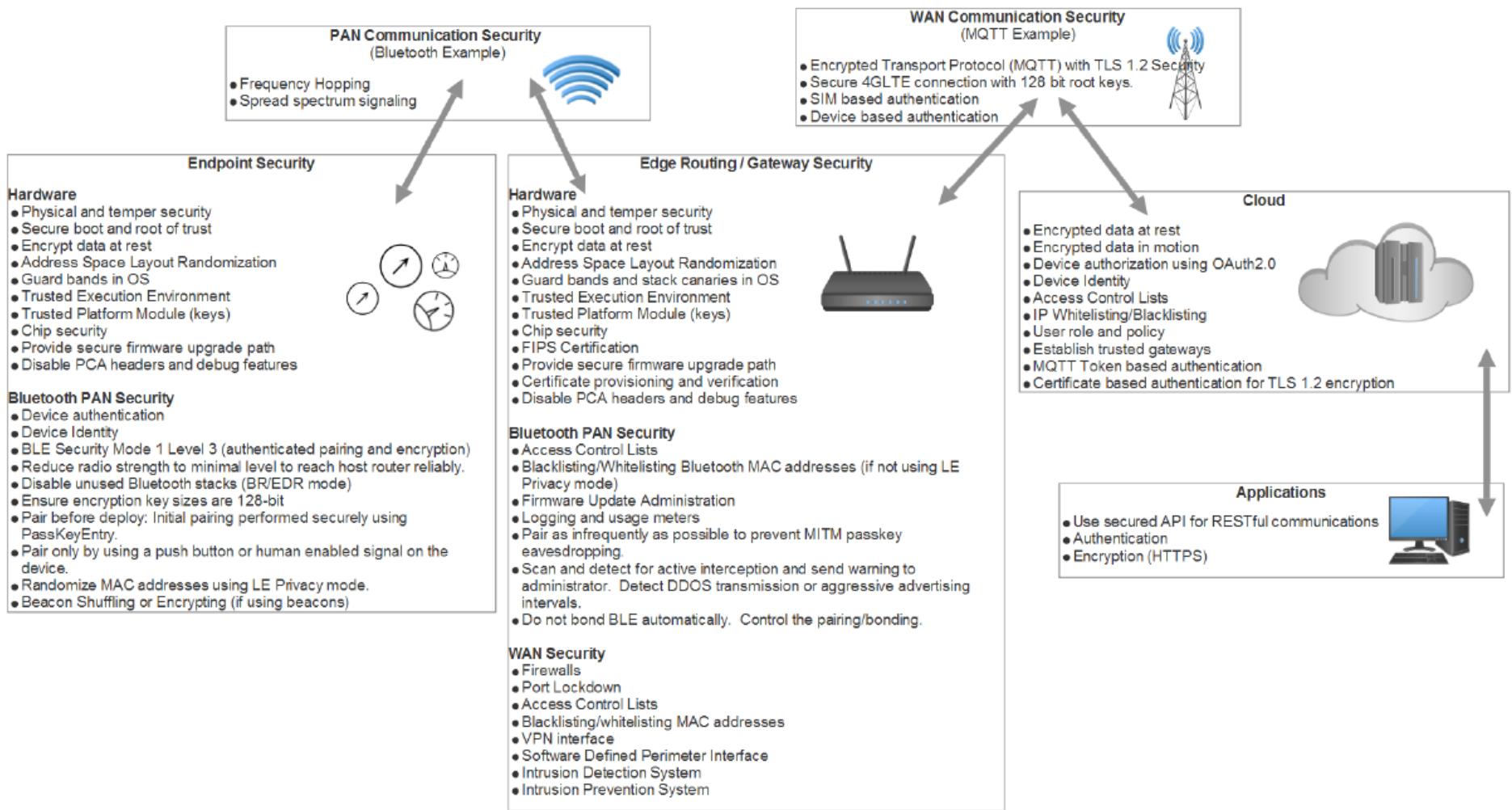
# Bitcoin blockchain



- Kriptovaluta namijenjena samo za IoT
- Sami IOT uređaji okosnica su mreže povjerenja, a arhitektura se temelji na usmjerenom acikličkom grafu (DAG)

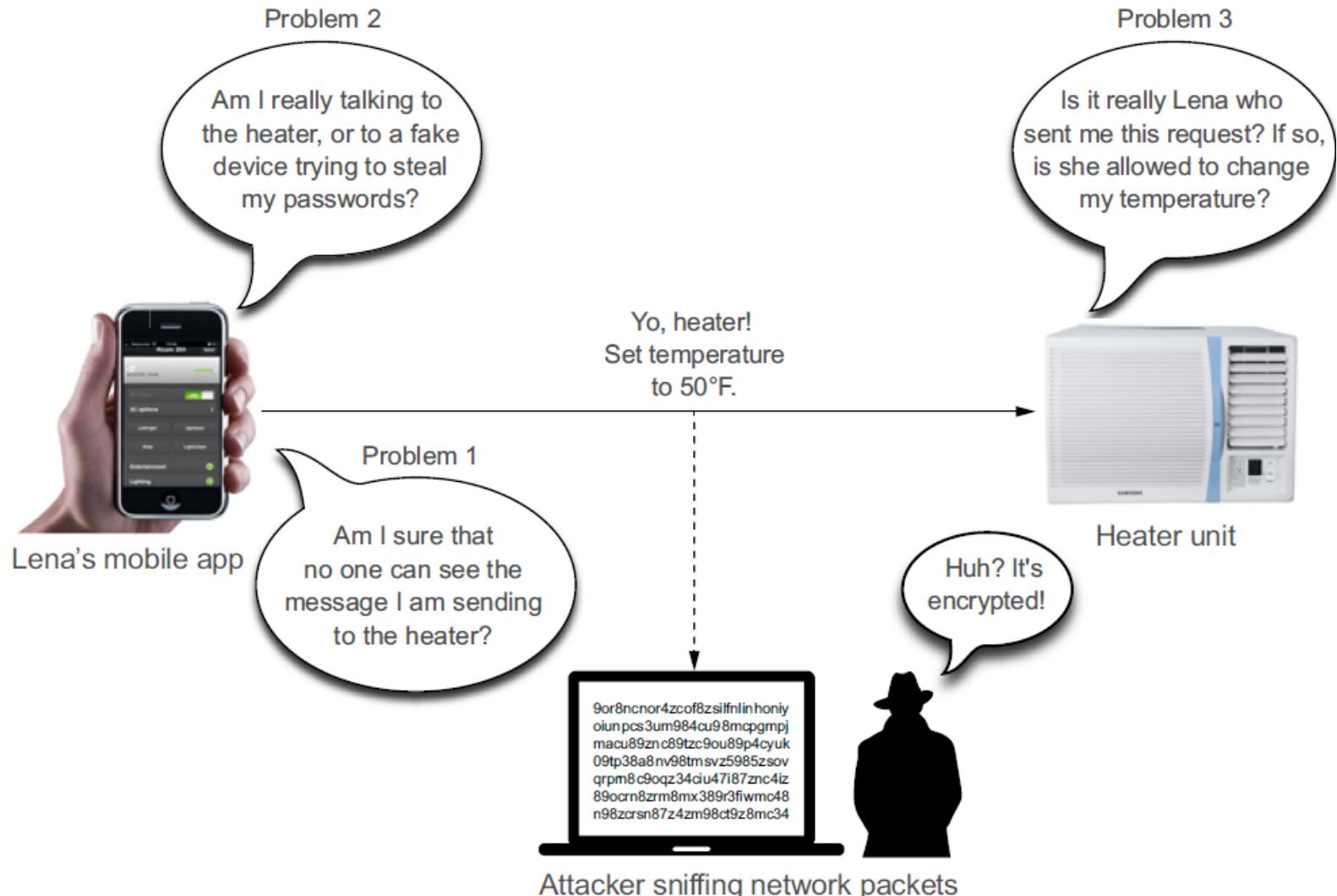
- IoT sigurnost također treba promatrati holistički od hardvera do oblaka
- Usko fokusiranje na jedan segment IoT-a ne pruža sigurnost i uspostavlja slabu kariku u sigurnosnom lancu.
- Potrebno je uspostaviti sigurnost od senzora do oblaka i natrag - holistički pristup.
- Svaka komponenta u lancu kontrole i podataka treba imati popis sigurnosnih parametara i pokretača.

# Primjer holističke sigurnosti u IoT-u

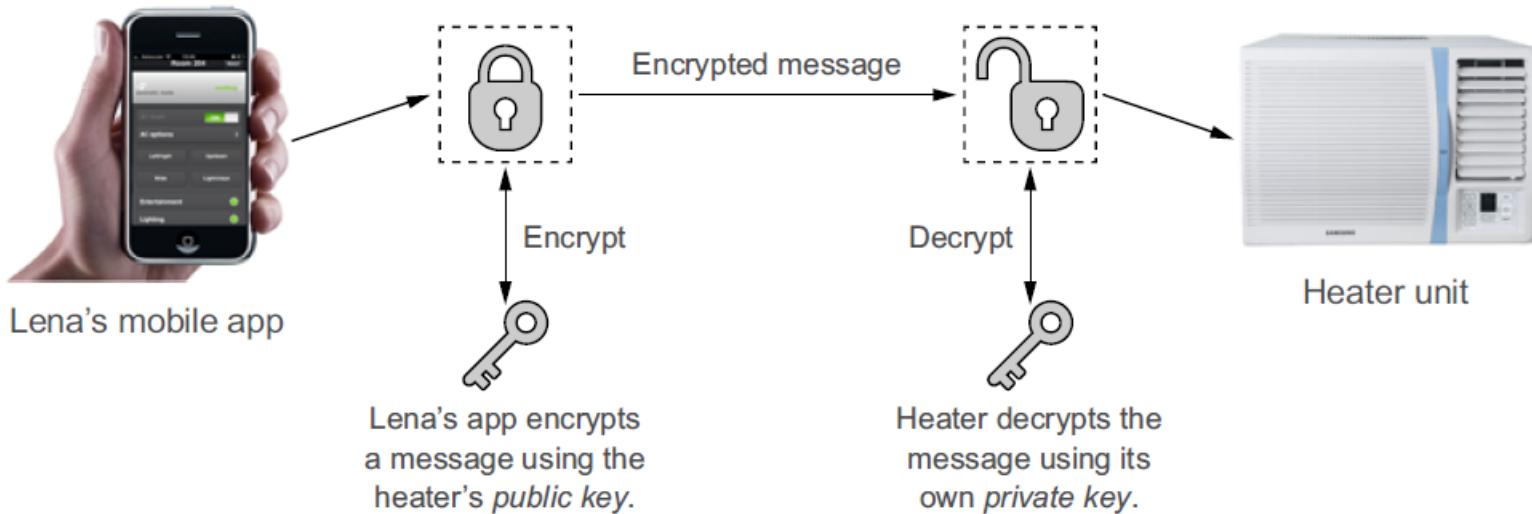


- Sigurnost na webu stvari još je kritičnija nego na webu.
- Budući da su web stvari fizički objekti koji će biti raspoređeni posvuda u stvarnom svijetu, rizici povezani s IoT napadima mogu biti katastrofalni.
- Većina IoT rješenja nije usklađena ni s najosnovnijim sigurnosnim najboljim praksama; sjetite se lozinki i komunikacija s jasnim tekstom, nevažećih certifikata, starih verzija softvera s greškama koje se mogu iskoristiti i tako dalje.
- Open Web Application Security Project (OWASP) Internet of Things projekt

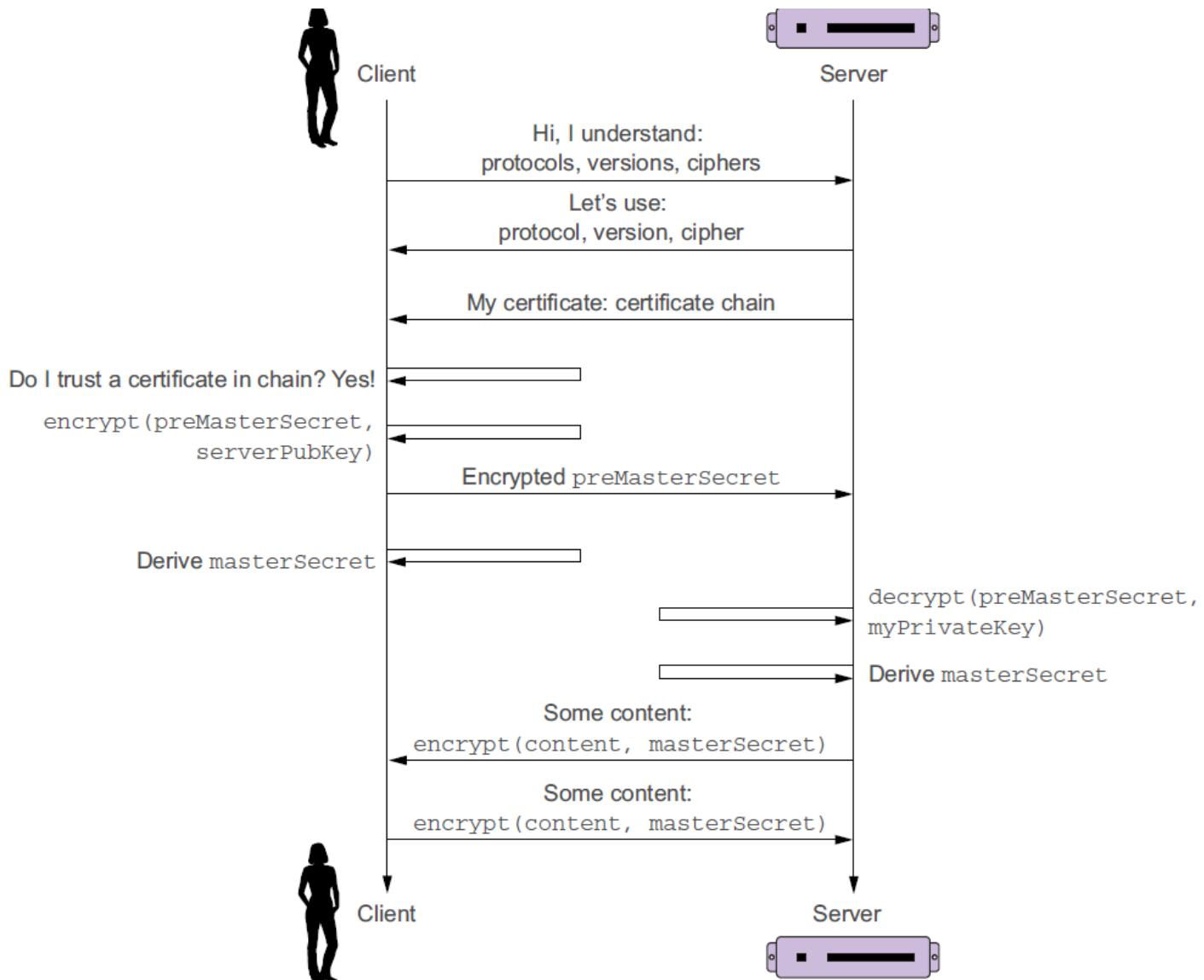
# Tri glavna sigurnosna problema u IoT-u



# Asimetrična enkripcija u IoT-u



# SSL/TLS rukovanje



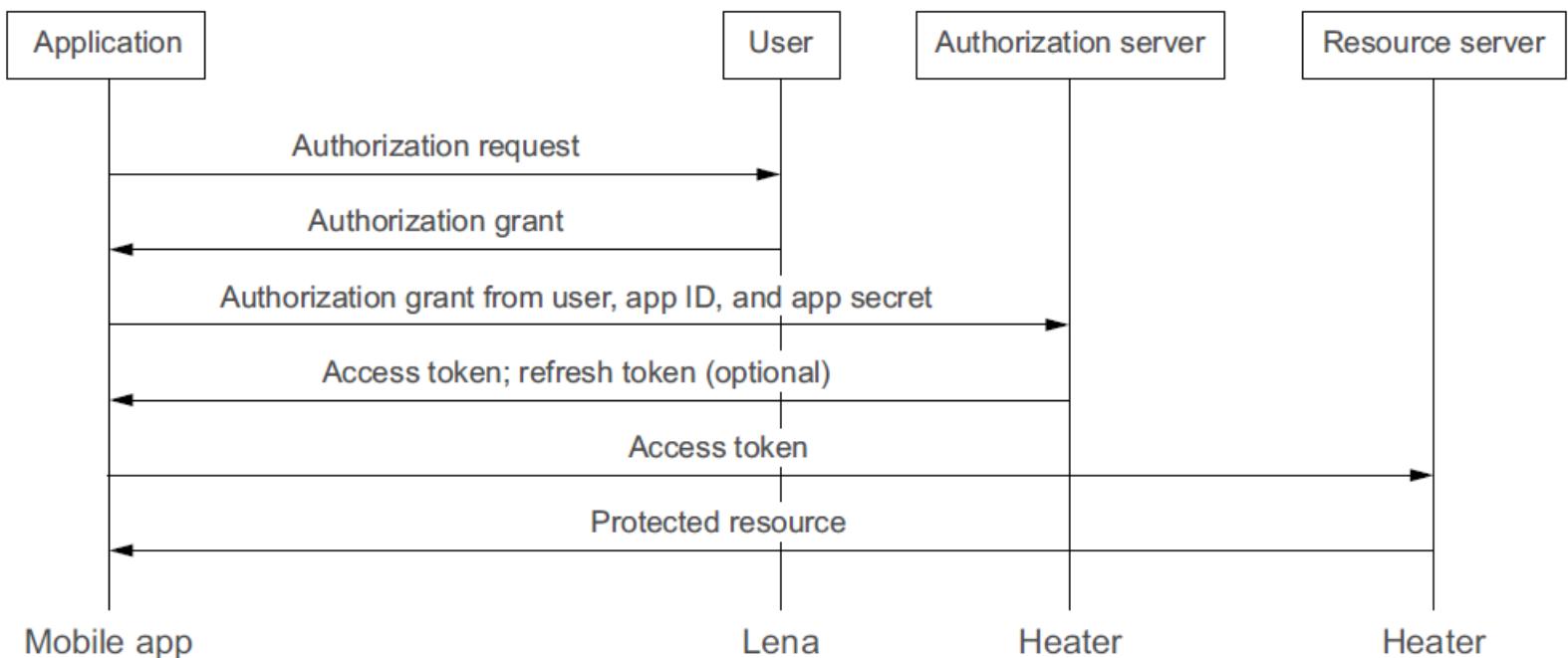
- Autentifikacija temeljena na tokenima
- Ideja je da se tajni token - dugačak niz znakova - koji je jedinstven za svakog klijenta, može koristiti za provjeru autentičnosti svakog zahtjeva koji taj klijent pošalje.
- Budući da se ovaj token dodaje zaglavljima ili parametrima upita svakog HTTP zahtjeva poslanog poslužitelju, sve interakcije ostaju bez stanja.
- Očito, API token treba generirati pomoću kriptografski sigurnog pseudo-slučajnog generatora i treba ga tretirati kao lozinku: pohraniti na šifriran način.
- API tokeni dobra su polazna točka, a uz enkripciju (TLS), nedvojbeno su minimum koji WoT uređaj treba ponuditi u smislu sigurnosti.

- Ali čim trebamo dijeliti resurse uređaja s nekoliko korisnika koji imaju različita autorizacijska prava, jednostavni API tokeni imaju probleme
- OAuth je otvoreni standard za autorizaciju i u biti je mehanizam za web ili mobilnu aplikaciju za delegiranje autentifikacije korisnika pouzdanoj usluzi treće strane
- Ukratko, OAuth standardizira kako autenticirati korisnike, generirati tokene s datumom isteka, ponovno generirati tokene i omogućiti pristup resursima na siguran i standardan način putem weba

## Uloge OAuth-a

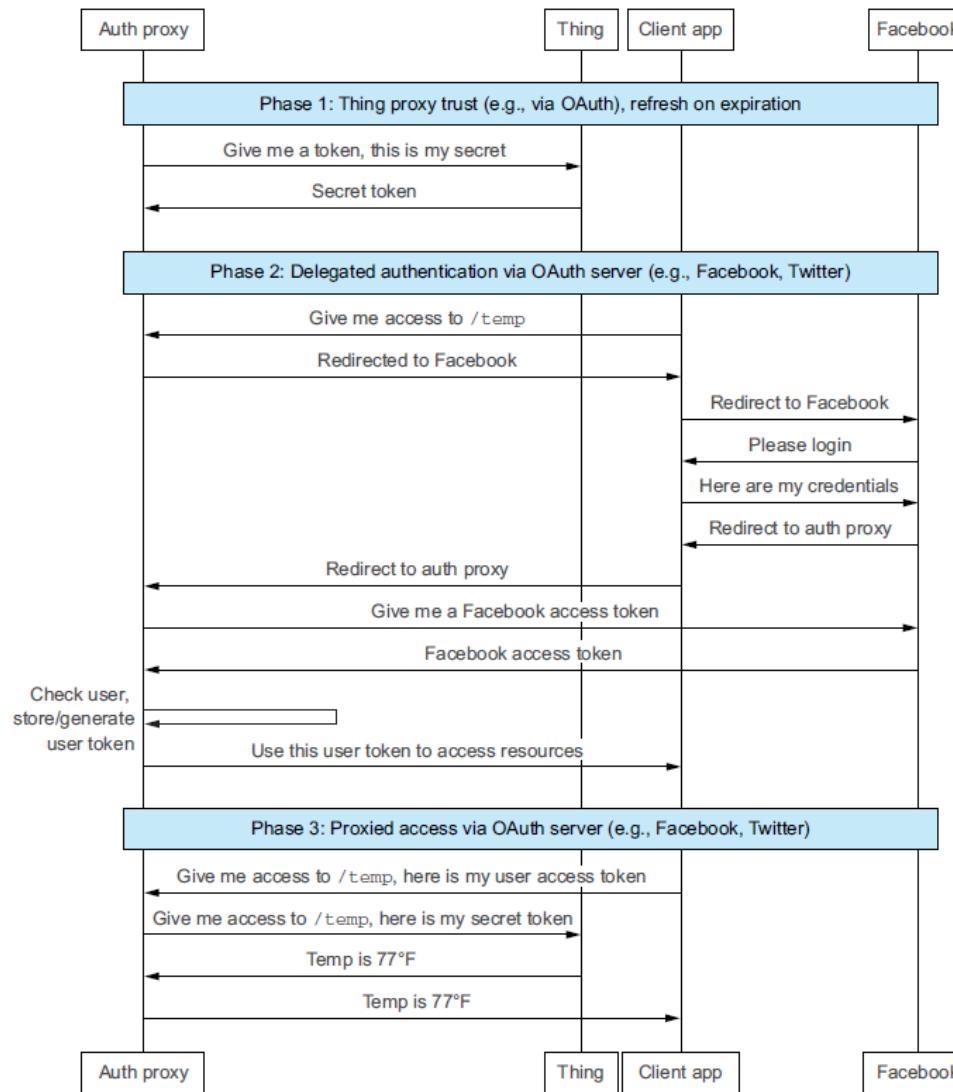
- Vlasnik resursa—ovo je korisnik koji želi autorizirati aplikaciju za pristup jednom od njegovih pouzdanih računa
- Poslužitelj resursa—u biti, ovo je web API koji prihvaca OAuth tokene kao vjerodajnice.
- Poslužitelj za autorizaciju—Ovo je OAuth poslužitelj koji upravlja autorizacijama za pristup resursima. To je web poslužitelj koji nudi OAuth API za autentifikaciju i autorizaciju korisnika. U nekim slučajevima, poslužitelj resursa i poslužitelj za autorizaciju mogu biti isti, kao u slučaju Facebooka.
- Aplikacija—ovo je web ili mobilna aplikacija koja želi pristupiti resursima korisnika.

# OAuth autentifikacija

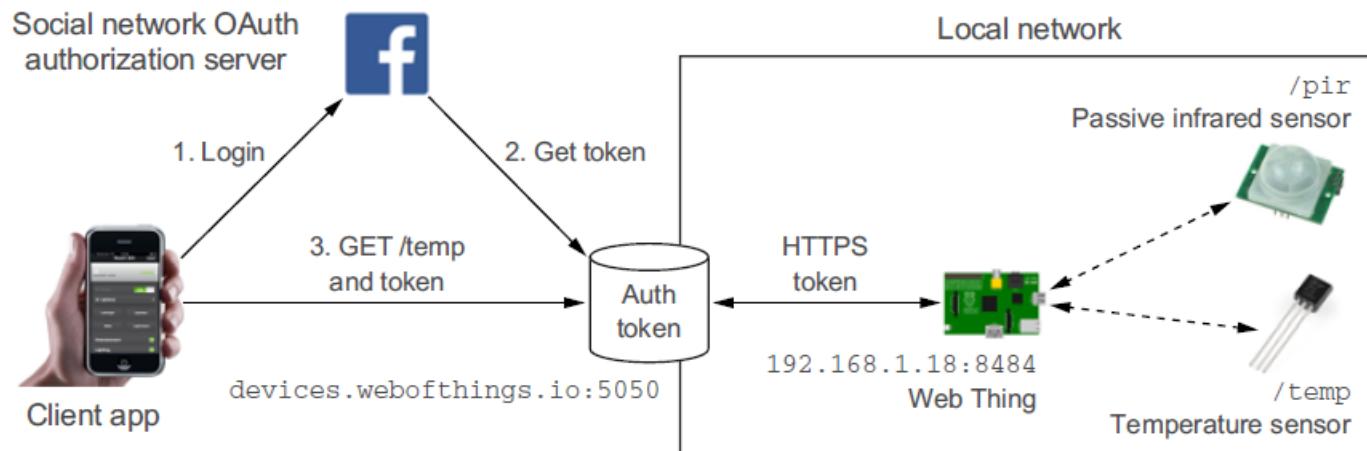


Iz knjige: Building the Web of Things with Examples in Node.js and Raspberry Pi

# Autentifikacijski proxy za društveni web stvari



# Implementacija autentifikacijskog proxy-ja društvenog weba stvari



Iz knjige: Building the Web of Things with Examples in Node.js and Raspberry Pi



# Analiza IoT podataka u oblaku i fogu

Darko Andročec

- Vrijednost IoT sustava nisu sami podaci iz jednog ili milijuna senzora, već interpretacija tih podataka i odluke koje se donose
- Analitika za IoT segment bavi se sljedećim vrstama podataka:
- Strukturirani podaci (SQL pohrana), predvidljivi format podataka.
- Nestrukturirani podaci (neobrađeni videopodaci ili signali), visok stupanj slučajnosti i varijance.
- Polustrukturirano (sažeci sadržaja na Twitteru), određeni stupanj varijance i slučajnosti u obliku.

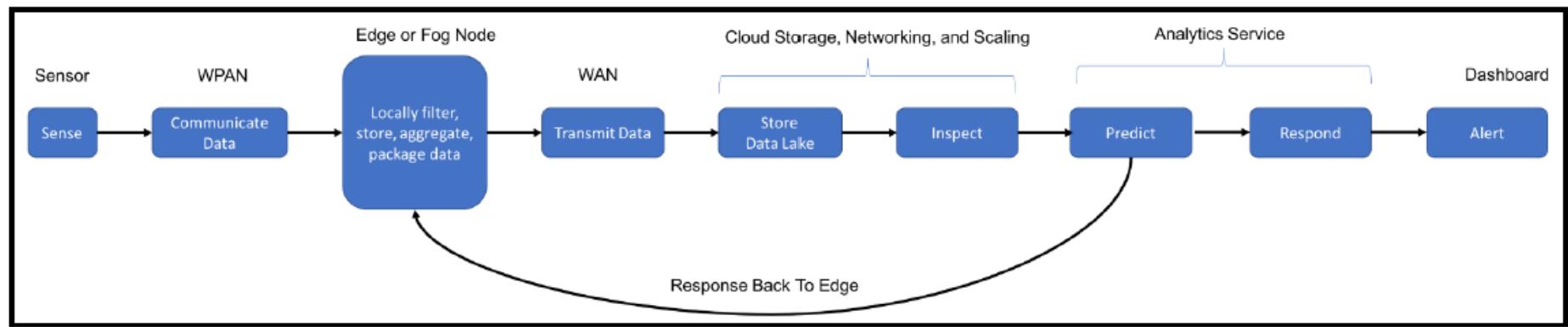
- Podatke će također možda trebati interpretirati i analizirati u stvarnom vremenu kao protok podataka ili se mogu arhivirati i dohvatiti za duboku analitiku u oblaku.
- Ovisno o slučaju upotrebe, podatke će možda trebati povezati s drugim izvorima
- U drugim slučajevima, podaci se jednostavno bilježe i odbacuju u podatkovno jezero kao što je Hadoop baza podataka.
- Zatim dolazi neka vrsta staginga, što znači da će sustav za razmjenu poruka kao što je Kafka usmjeravati podatke prema procesoru toka, ili batch procesoru, ili možda oboje.

- Predobrada (engl. preprocessing) - Filtriranje događaja od malog interesa, denaturacija, ekstrakcija značajki, segmentacija, transformiranje podataka u prikladniji oblik (iako podatkovna jezera ne preferiraju trenutnu transformaciju), dodavanje atributa podacima kao što je oznaka (podatkovna jezera trebaju oznake).
- Uzbunjivanje (engl. alerting) - Pregledajte podatke; ako premaši neki granični uvjet, onda pokrenite upozorenje. Najjednostavniji primjer je ako temperatura poraste iznad postavljene granice na senzoru.
- Rad s prozorima (engl. windowing) - Dobro za pravila i za brojanje događaja. Moglo bi se potražiti broj skokova temperature u posljednjem satu i zaključiti da će se na nekom stroju pojaviti kvar.

- Pridruživanje (engl. joins) - Kombiniranje više tokova podataka u jedan novi tok.
- Greške (engl. errors) - Milijuni senzora će generirati podatke koji nedostaju, iskrivljene podatke i podatke koji nisu u nizu.
- Vremenski događaji i predlošci – npr. otkrivanje da li je temperatura prešla 100 °C
- Praćenje (engl. tracking) - Praćenje uključuje kada ili gdje nešto postoji, dogodio se događaj ili kada nešto ne postoji tamo gdje je trebalo – npr. praćenje lokacija kamiona
- Trendovi - Tekuća povijest vremenski koreliranih podataka mogla bi se koristiti za pronalaženje obrazaca poput senzora za stoku u poljoprivredi

- Skupni upiti (engl. batch queries) – npr. Lambda obrada
- Put duboke analitike (engl. Deep analytics pathway) – npr. sustav za video nadzor
- Modeli i treniranje - Mehanizam za zaključivanje za sustav strojnog učenja. Ovi alati za strojno učenje izgrađeni su na obučenim modelima koji se mogu koristiti za analizu u stvarnom vremenu.
- Signalizacija - Na primjer, ako temperatura poraste iznad određene granice na stroju, zabilježite događaj, ali također pošaljite signal rubnom uređaju da uspori stroj.
- Kontrola - Konačno, trebamo način za kontrolu ovih alata za analizu. Bilo da se radi o pokretanju, zaustavljanju, izvješćivanju, bilježenju ili otklanjanju pogrešaka, moraju postojati sredstva za upravljanje ovim sustavom.

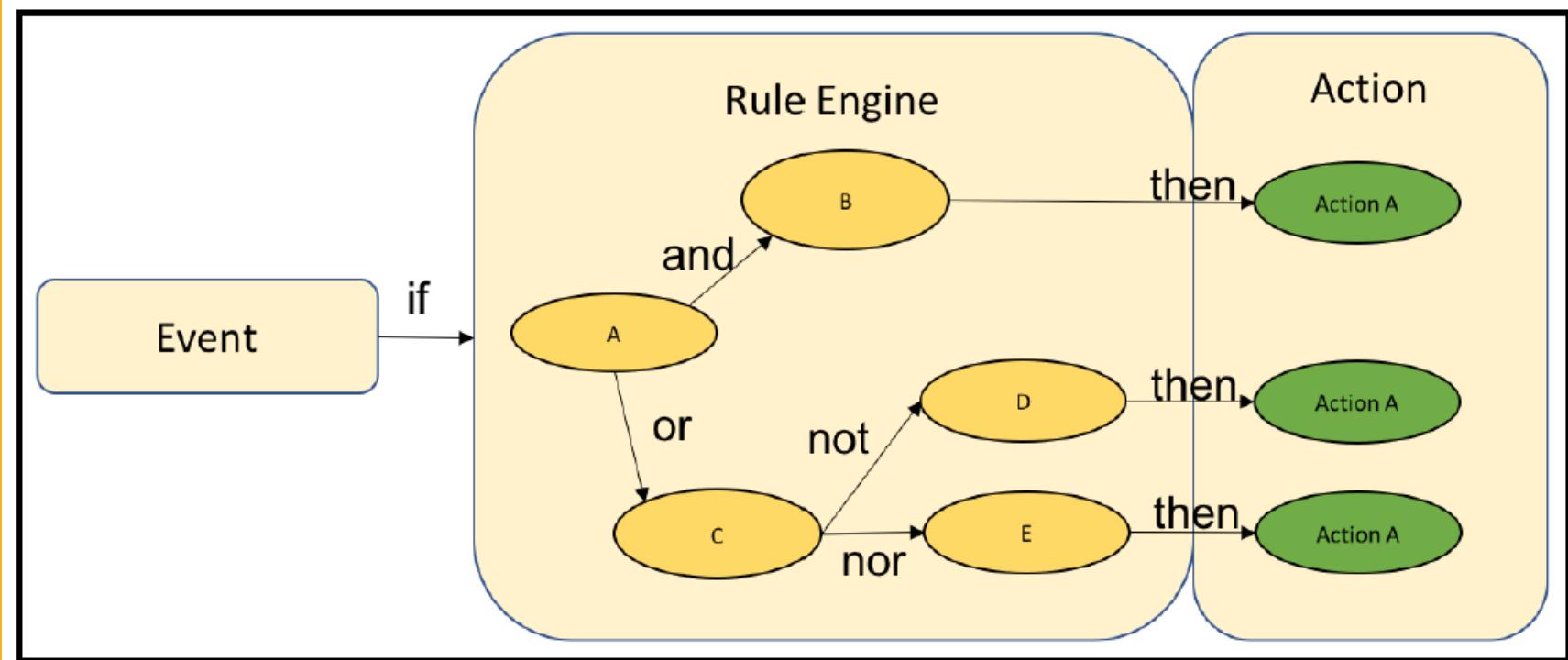
# Cjevod oblaka (cloud pipeline)



Iz knjige: Internet of Things for Architects

- Softver konstruiran da izvede akcije prema događajima
- Npr., ako vlažnost zraka u prostoriji pređe 50%, pošalji SMS poruku vlasniku
- Zovu se i sustavi za upravljanje poslovnim pravilima (engl. Business Rule Management Systems)
- Mogu ali i ne moraju pamtitи stanje
- Može imati povijest događaja i poduzimati različite radnje ovisno o redoslijedu, količini ili obrascima događaja kako su se događali u povijesti.
- Alternativno, možda neće održavati stanje i samo može pregledavati trenutni događaj

# Rule engine - primjer

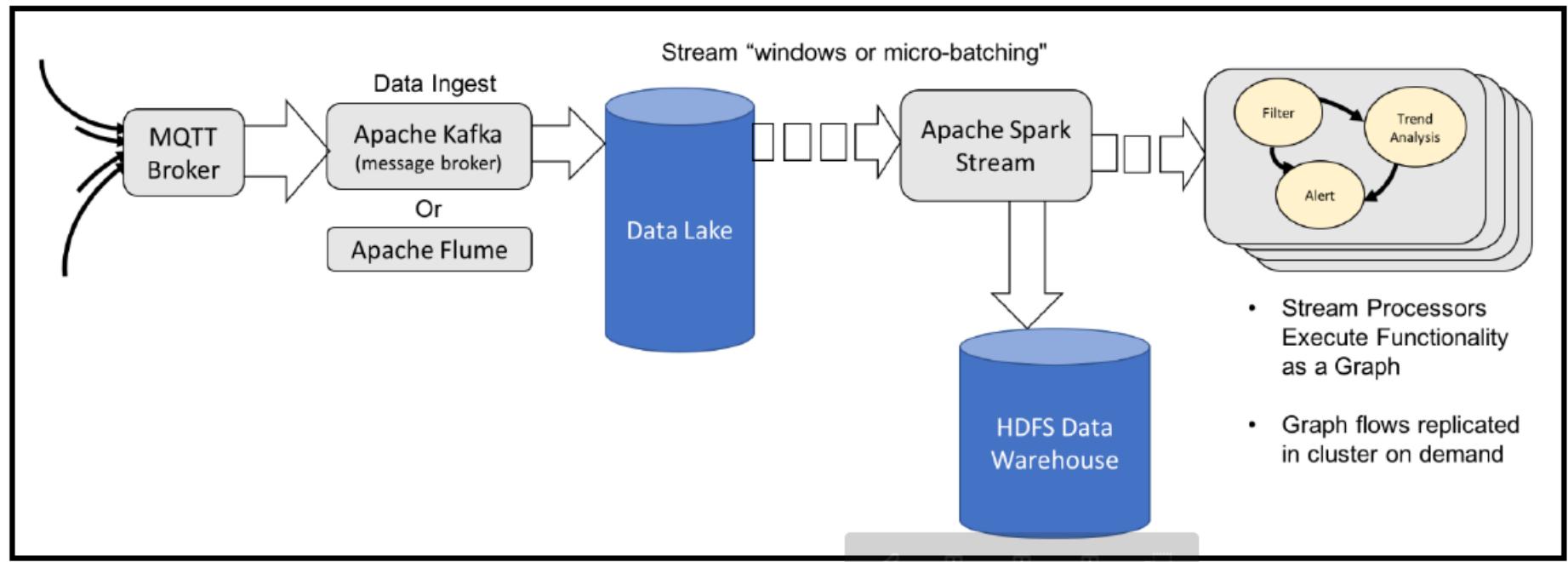


Iz knjige: Internet of Things for Architects

- <https://www.drools.org/>

```
Smoke Sensor = Smoke Detected
Heat Sensor = Heat Detected
if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor ==
Heat_Detected) then
    Fire
    if (Smoke_Sensor == !Smoke_Detected) && (Heat_Sensor ==
Heat_Detected) then
        Furnace_On
        if (Smoke_Sensor == Smoke_Detected) && (Heat_Sensor ==
!Heat_Detected) then
            Smoking
            if (Fire) then Alarm
            if (Furnace_On) then Log_Temperature
            if (Smoking) then SMS_No_Smoking_Allowed
```

# Unos s oblaka u skladište podataka

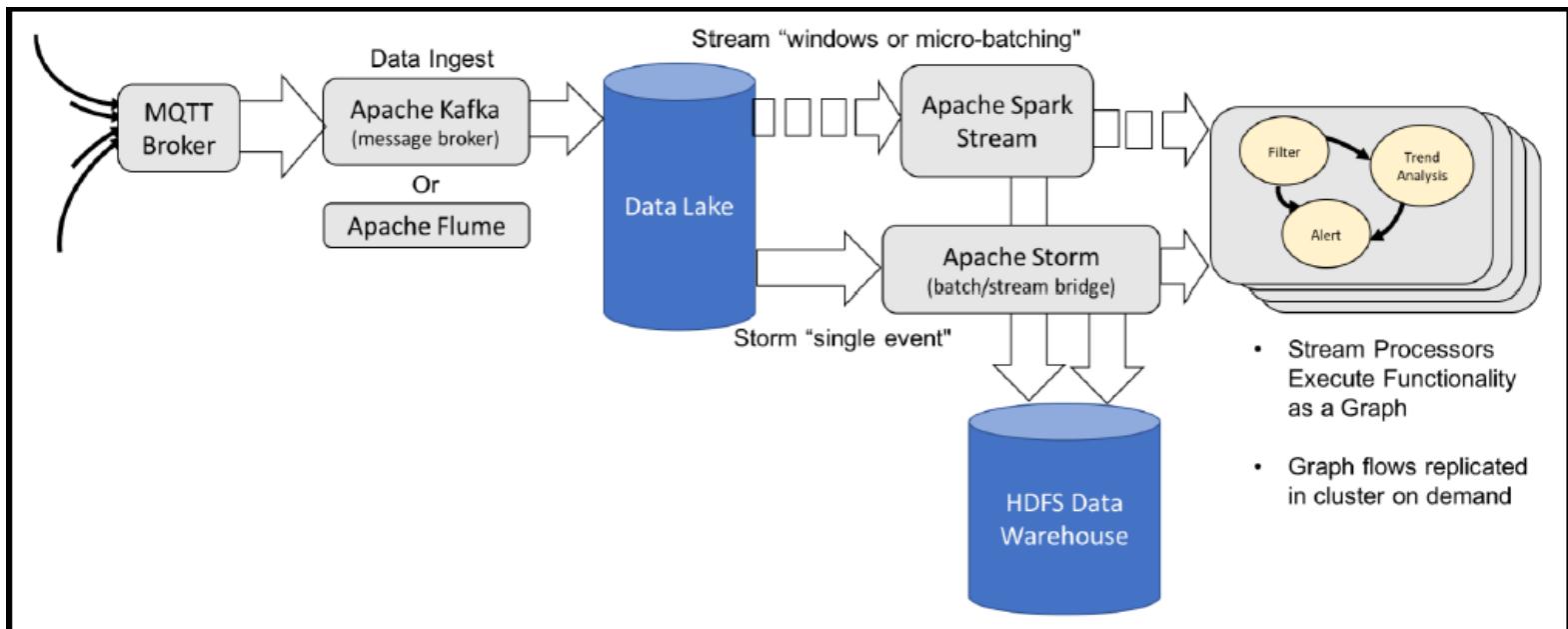


Iz knjige: Internet of Things for Architects

- Complex event processing (CEP)
- CEP sustavi koriste upite slične SQL-u, ali umjesto da koriste pozadinu baze podataka, pretražuju dolazni tok za uzorak ili pravilo koje predlažete.
- Diskretni podatkovni element s vremenskom oznakom
- Primjer: Apache WSO2 CEP

# Lambda arhitektura

- Lambda arhitektura pokušava uravnotežiti latenciju s propusnošću.
- U suštini, miješa seriju s obradom toka.
- Slično općoj topologiji oblaka OpenStacka ili drugih okvira oblaka, Lambda unosi i pohranjuje u nepromjenjivo spremište podataka.

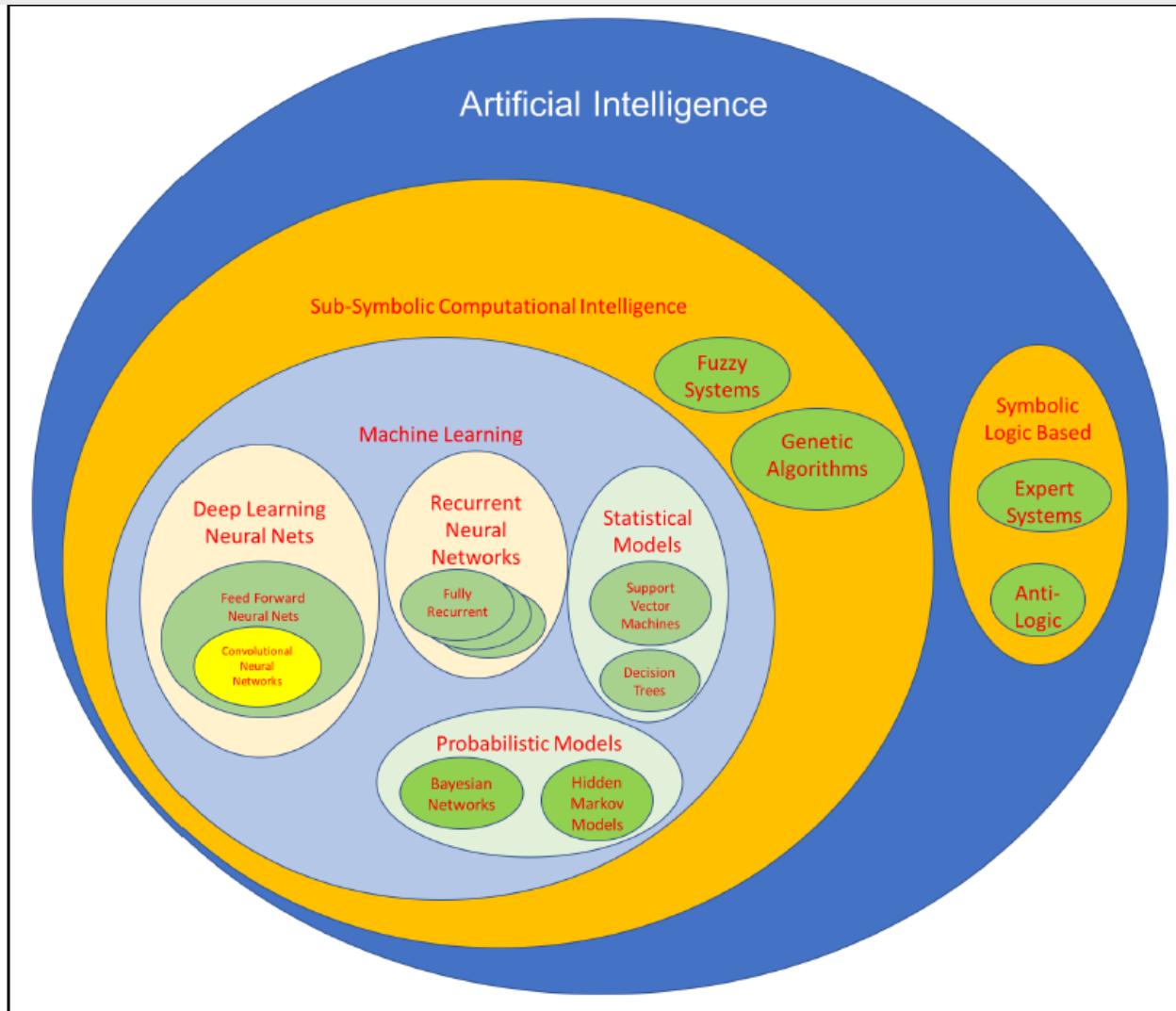


# Primjeri slučajeva korištenja analitike u oblaku za IoT



Industry	Use cases	Cloud services	Typical bandwidth	Real time	Analytics
Manufacturing	<ul style="list-style-type: none"> <li>Operational technology</li> <li>Brownfield</li> <li>Asset tracking</li> <li>Factory automation</li> </ul>	<ul style="list-style-type: none"> <li>Dashboards</li> <li>Bulk storage</li> <li>Data lakes</li> <li>SDN (hybrid cloud topology)</li> <li>Low latency</li> </ul>	<ul style="list-style-type: none"> <li>500 GB/day/factory part produced</li> <li>2 TB/minute mining operations</li> </ul>	Less than 1s	<ul style="list-style-type: none"> <li>Recurrent neural nets</li> <li>Bayesian networks</li> </ul>
Logistics and transport	<ul style="list-style-type: none"> <li>Geolocation tracking</li> <li>Asset tracking</li> <li>Equipment sensing</li> </ul>	<ul style="list-style-type: none"> <li>Dashboards</li> <li>Logging</li> <li>Storage</li> </ul>	<ul style="list-style-type: none"> <li>Vehicles: 4 TB/day/vehicle (50 sensors)</li> <li>Aircraft: 2.5 to 10 TB/day (6000 sensors)</li> <li>Assets tracking: 1 MB/day/beacon</li> </ul>	<ul style="list-style-type: none"> <li>Less than 1s (real-time)</li> <li>Daily (batch)</li> </ul>	Rule engines
Healthcare	<ul style="list-style-type: none"> <li>Asset tracking</li> <li>Patient tracking</li> <li>Home health monitoring</li> <li>Wireless health equipment</li> </ul>	<ul style="list-style-type: none"> <li>Reliability and HIPPA</li> <li>Private cloud option</li> <li>Storage and archival</li> <li>Load balancing</li> </ul>	<ul style="list-style-type: none"> <li>1 MB/day/sensor</li> </ul>	<ul style="list-style-type: none"> <li>Less than 1s: Life critical</li> <li>Non-life critical: On each change</li> </ul>	<ul style="list-style-type: none"> <li>Recurrent Neural Networks (RNN)</li> <li>Decision trees</li> <li>Rules engines</li> </ul>
Agriculture	<ul style="list-style-type: none"> <li>Livestock health and location tracking</li> <li>Soil chemistry analysis</li> </ul>	<ul style="list-style-type: none"> <li>Bulk storage - archiving</li> <li>Cloud-to-cloud provisioning</li> </ul>	<ul style="list-style-type: none"> <li>512 KB/day/livestock head</li> <li>1000 to 10000 head of cattle per feedlot</li> </ul>	<ul style="list-style-type: none"> <li>1 second (real-time)</li> <li>10 minutes (batch)</li> </ul>	Rules engines
Energy	<ul style="list-style-type: none"> <li>Smart meters</li> <li>Remote energy monitoring (solar, natural gas, oil)</li> <li>Failure prediction</li> </ul>	<ul style="list-style-type: none"> <li>Dashboards</li> <li>Data lakes</li> <li>Bulk storage for Historical rate prediction</li> <li>SDN</li> <li>Low latency</li> </ul>	<ul style="list-style-type: none"> <li>100-200 GB/day/wind turbine</li> <li>1 to 2 TB/day/oil rig</li> <li>100 MB/day/smart meter</li> </ul>	<ul style="list-style-type: none"> <li>Less than 1s: energy production</li> <li>1 minute: smart meters</li> </ul>	<ul style="list-style-type: none"> <li>RNN</li> <li>Bayesian networks</li> <li>Rules engines</li> </ul>
Consumer	<ul style="list-style-type: none"> <li>Real-time health logging</li> <li>Presence detection</li> <li>Lighting and heating/AC</li> <li>Security</li> <li>Connected home</li> </ul>	<ul style="list-style-type: none"> <li>Dashboards</li> <li>PaaS</li> <li>Load balancing</li> <li>Bulk storage</li> </ul>	<ul style="list-style-type: none"> <li>Security camera: 500 GB/day/camera</li> <li>Smart device: 1-1000 KB/day/sensor-device</li> <li>Smart home: 100 MB/day/home</li> </ul>	<ul style="list-style-type: none"> <li>Video: less than 1s</li> <li>Smart home: 1s</li> </ul>	<ul style="list-style-type: none"> <li>Convolutional neural nets (image sensing)</li> <li>Rules engines</li> </ul>
Retail	<ul style="list-style-type: none"> <li>Cold chain sensing</li> <li>POS machines</li> <li>Security systems</li> <li>Beaconing</li> </ul>	<ul style="list-style-type: none"> <li>SDN/SDP</li> <li>Micro-segmentation</li> <li>Dashboards</li> </ul>	<ul style="list-style-type: none"> <li>Security: 500 GB/day/camera</li> <li>General: 1-1000 MB/day/device</li> </ul>	<ul style="list-style-type: none"> <li>POS and credit transaction: 100ms</li> <li>Beaconing: 1s</li> </ul>	<ul style="list-style-type: none"> <li>Rules engines</li> <li>Convolutional neural networks for security</li> </ul>
Smart City	<ul style="list-style-type: none"> <li>Smart parking</li> <li>Smart trash pickup</li> <li>Environmental sensors</li> </ul>	<ul style="list-style-type: none"> <li>Dashboards</li> <li>Data lakes</li> <li>Cloud-to-cloud services</li> </ul>	<ul style="list-style-type: none"> <li>Energy monitors: 2.5 GB/day/city (70K sensors)</li> <li>Parking spots: 300 MB/day (80,000 sensors)</li> <li>Waste monitors: 350 MB/day (200,000 sensors)</li> <li>Noise monitors: 650 MB/day (30,000 sensors)</li> </ul>	<ul style="list-style-type: none"> <li>Electric meters: 1 minute</li> <li>Temperature: 15 minutes</li> <li>Noise: 1 minute</li> <li>Waste: 10 minutes</li> <li>Parking spots: every change</li> </ul>	<ul style="list-style-type: none"> <li>Rules engine</li> <li>Decision trees</li> </ul>

# Algoritmi umjetne inteligencije



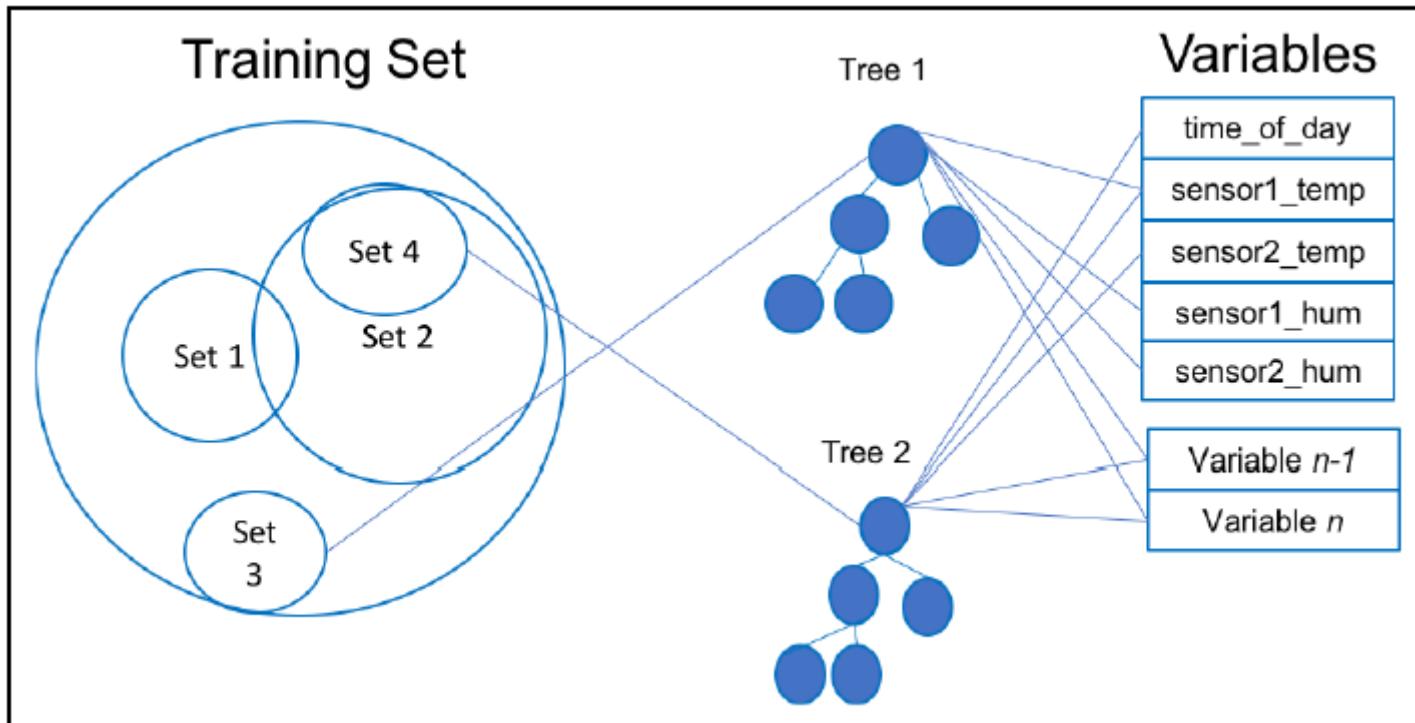
- IoT – ogromna količina neprekinutih podataka (engl. streaming data)
- Vrijednost sustava senzora nije ono što jedan senzor mjeri, već ono što skup senzora mjeri i govori nam o mnogo većem sustavu.
- Neki od tih podataka bit će strukturirani: vremenski korelirane serije.
- Ostali podaci bit će nestrukturirani: kamere, sintetički senzori, audio i analogni signali.
- Klijent na temelju tih podataka želi donijeti korisne odluke za svoje poslovanje.

- Mogla bi se promatrati cijela tvornica kako bi se razumjelo kako ta tvornica radi u tom trenutku na temelju zbirke milijuna ili milijardi događaja sa svakog stroja i svakog radnika u tom sustavu.
- S tom količinom podataka, samo uređaj za strojno učenje može izbjegći šum i pronaći relevantnost.
- To nisu problemi s kojima se ljudi mogu nositi, već problemi velikih podataka (engl. Big Data) i strojnog učenja.

- Nadzirano učenje (engl. *supervised learning*) - Podaci za treniranje (engl. training dana) dani modelu imaju pridruženu oznaku sa svakim unosom - omogućuje rješavanje problema klasifikacije i regresije.
- Nenadzirano učenje (engl. *unsupervised learning*) – Nema labele za podatke za treniranje. - Ova vrsta modela učenja koristi matematička pravila za smanjenje redundantnosti. Tipičan slučaj upotrebe je pronalaženje klastera sličnih stvari.

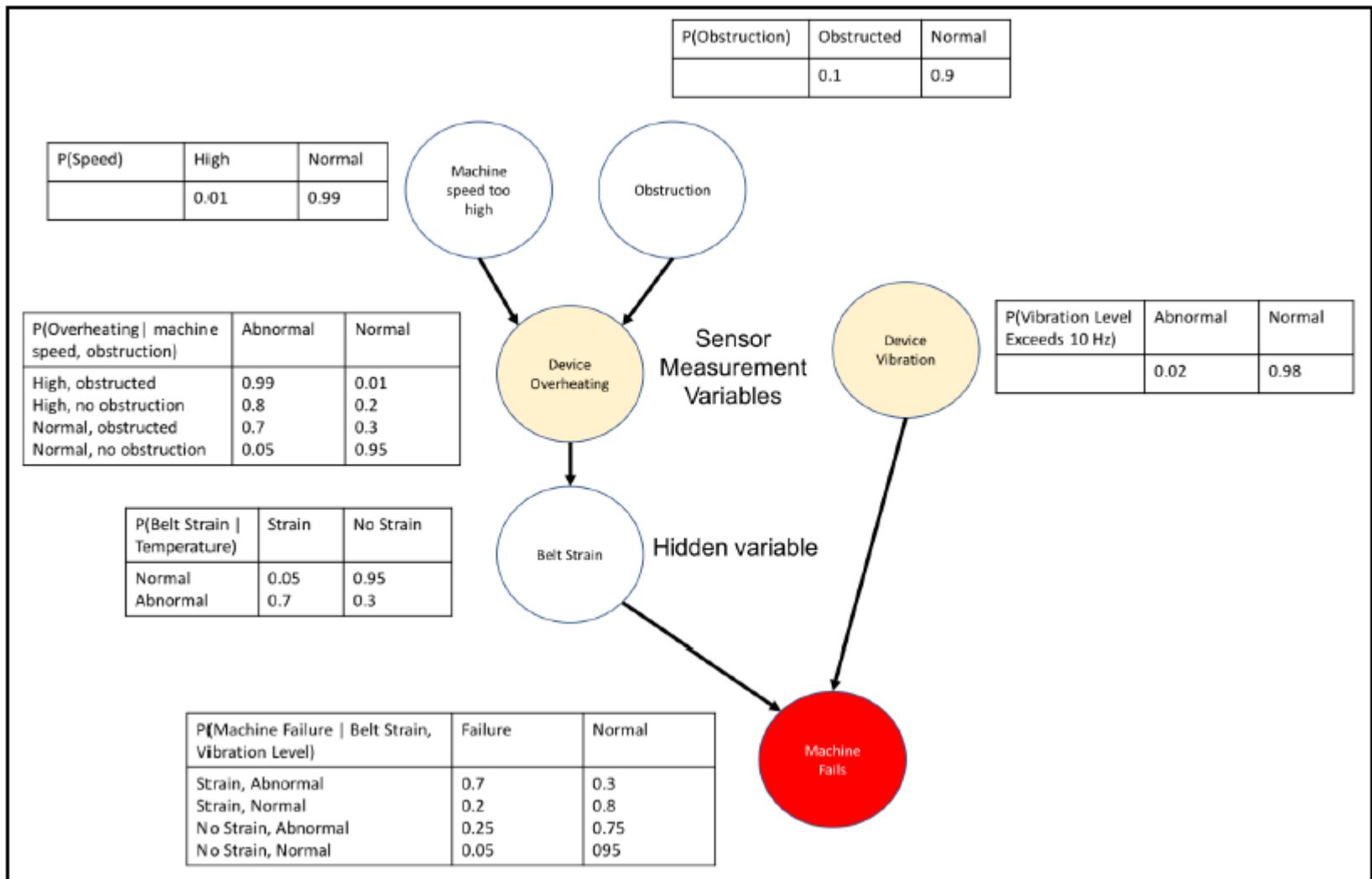
- Klasifikacija
- Regresija
- Pronalaženje anomalija

# Random forest



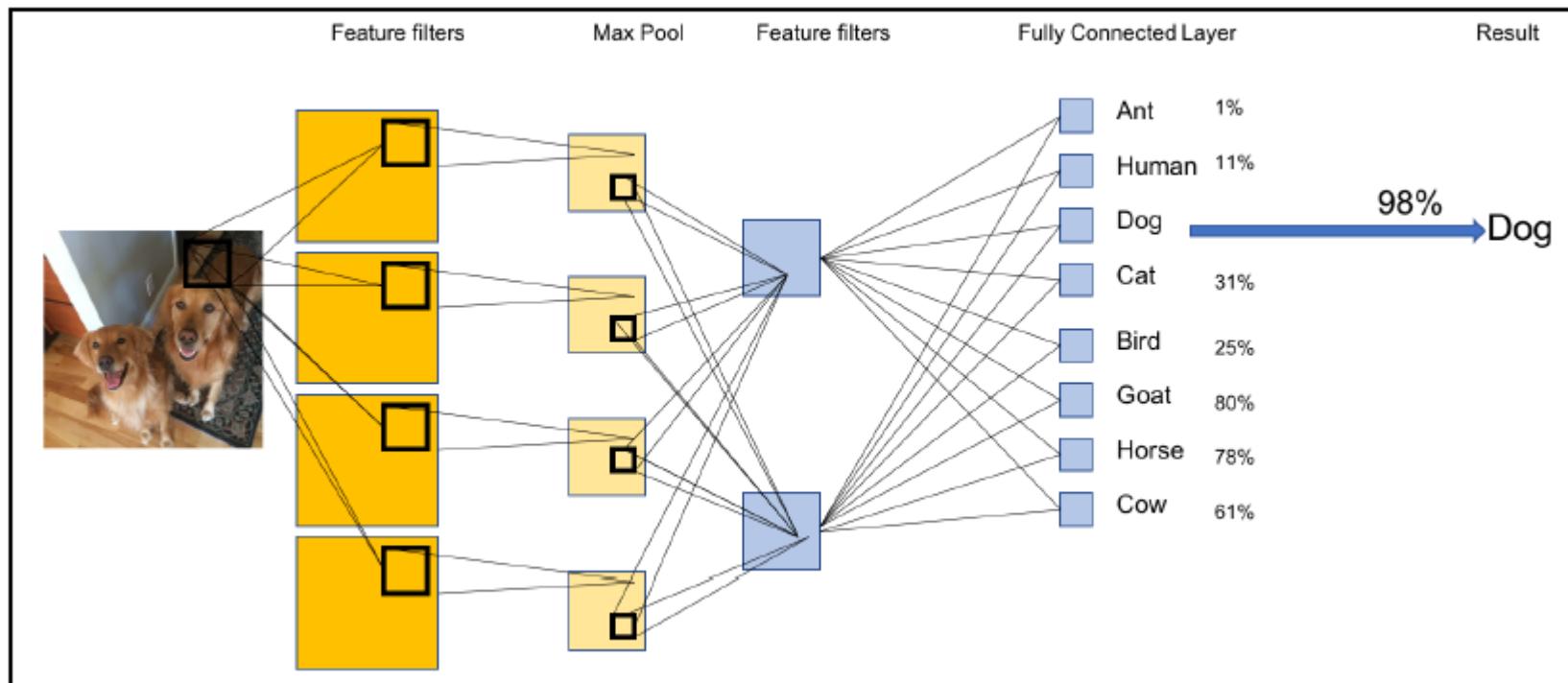
Iz knjige: Internet of Things for Architects

# Bayesovi modeli



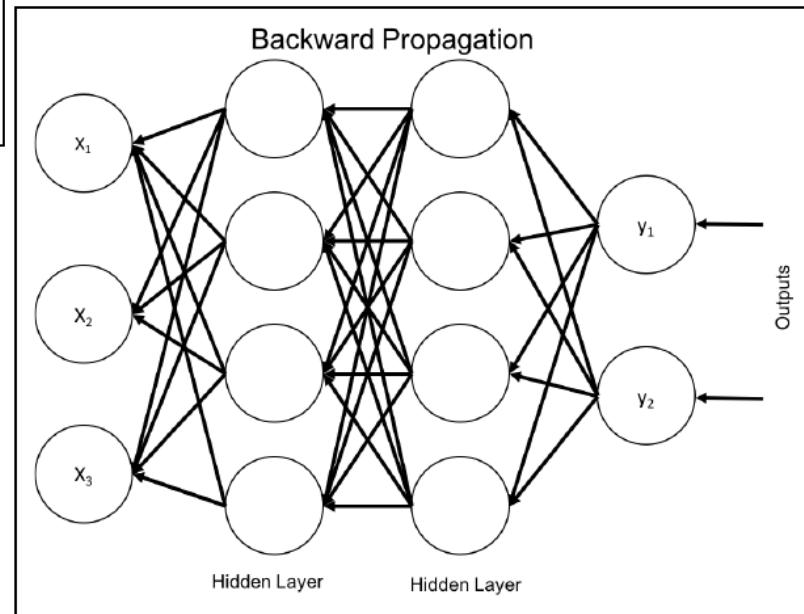
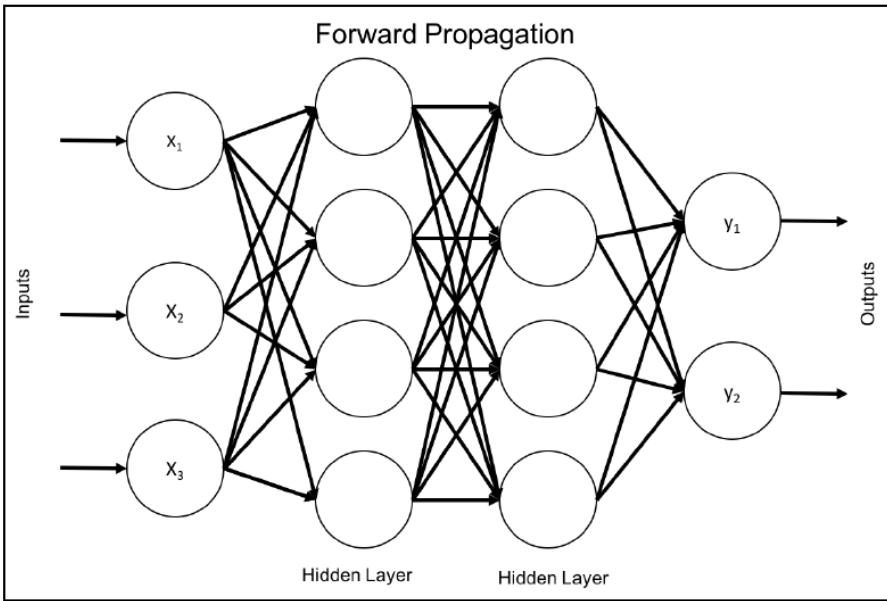
# CNN (Convolutional Neural Networks)

- Vrlo pouzdan i točan u klasifikaciji slika



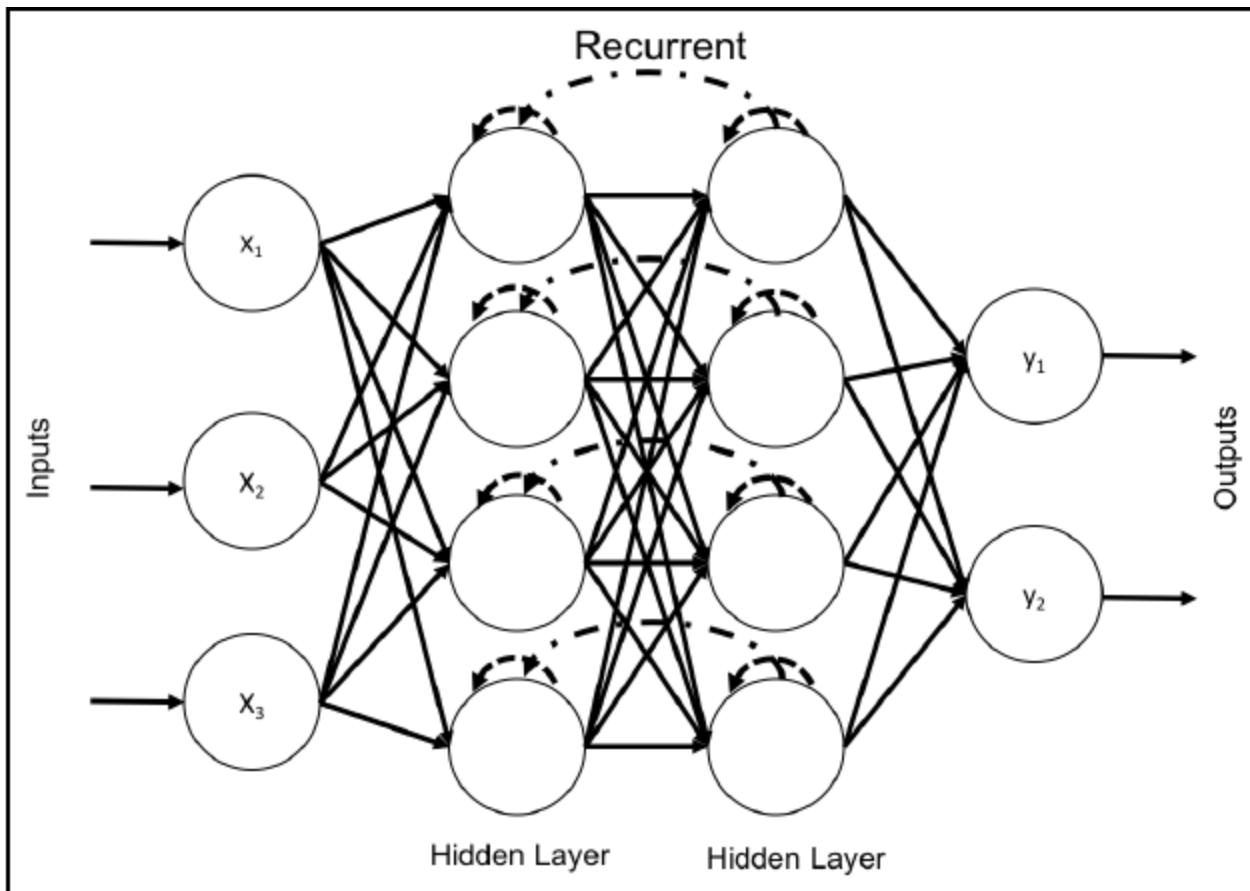
Iz knjige: Internet of Things for Architects

# Treniranje CNN-a



- Važni za rad s IoT podacima
- RNN razumije vremensku prirodu stvari, čuva stanje.
- RNN-ovi su od posebne vrijednosti u IoT prostoru, posebno u vremenski koreliranim nizovima podataka, kao što je opisivanje scene na slici, opisivanje osjećaja (engl. sentiment) niza teksta ili vrijednosti i klasificiranje video tokova.

## RNN (2)



Iz knjige: Internet of Things for Architects

- U IoT svijetu kašnjenje/latencija je veliki problem, posebno za sigurnosno kritičnu infrastrukturu.
- Ograničenja resursa još su jedan faktor. Većina današnjih rubnih računalnih uređaja nema na raspolaganju hardverske akceleratore kao što su General-Purpose Computation on Graphics Hardware (GPGPU) i Field-programmable gate array (FPGA) koji bi pomogli u složenoj matričnoj matematici i pokretnom zarezu koji trebaju za neuronske mreže.
- Podaci se mogu slati u oblak, ali to može imati značajne učinke kašnjenja kao i troškove propusnosti.

# Primjeri IoT analitike podataka

Model	Best application	Worse fit and side effects	Resource demands	Training
Random forests (statistical models)	<ul style="list-style-type: none"> <li>Anomaly detection</li> <li>Systems with 1000's of choice points and hundreds of inputs</li> <li>Regression and classification</li> <li>Handles mixed data types</li> <li>Ignores missing values</li> <li>Scales linearly with input</li> </ul>	<ul style="list-style-type: none"> <li>Feature extraction</li> <li>Time and sequence analysis</li> </ul>	Low	<ul style="list-style-type: none"> <li>Training based on bagging techniques. for maximum effectiveness</li> <li>Training fairly resource light</li> <li>Mainly supervised</li> </ul>
RNN (temporal and sequence-based neural networks)	<ul style="list-style-type: none"> <li>Prediction of an event based on a sequence</li> <li>Streaming data patterns</li> <li>Time-correlated series data</li> <li>Maintains knowledge of past states to predict new states (electrical signals, audio, speech recognition)</li> <li>Unstructured data</li> <li>Input variables may or may not be dependent</li> </ul>	<ul style="list-style-type: none"> <li>Image and video analysis</li> <li>Systems of requiring thousands of features</li> </ul>	<ul style="list-style-type: none"> <li>Very high for training</li> <li>High for inference execution</li> </ul>	<ul style="list-style-type: none"> <li>Training more cumbersome than CNN backpropagation</li> <li>Very hard to train</li> <li>Supervised</li> </ul>
CNN (deep learning)	<ul style="list-style-type: none"> <li>Prediction of an object based on surrounding values</li> <li>Pattern and feature identification</li> <li>2D image recognition</li> <li>Unstructured Data</li> <li>Input variables may or may not be dependent</li> </ul>	<ul style="list-style-type: none"> <li>Time-based and sequential predictions</li> <li>Systems of requiring thousands of features</li> </ul>	<ul style="list-style-type: none"> <li>Very high for Training (floating point precision, large training sets, large memory demands)</li> <li>High for inference execution</li> </ul>	Supervised and unsupervised
Bayesian networks (probabilistic models)	<ul style="list-style-type: none"> <li>Noisy and incomplete data sets</li> <li>Streaming data patterns</li> <li>Time correlated series</li> <li>Structured data</li> <li>Signal analysis</li> <li>Models developed quickly</li> </ul>	<ul style="list-style-type: none"> <li>Assumes all input variables are independent</li> <li>Perform poorly with high orders of data dimensions</li> </ul>	Low	<ul style="list-style-type: none"> <li>Little training data need with respect to other artificial neural networks</li> </ul>

