Veebiteenuste ja hajussüsteemide arendus

Loeng 3: Teadete edastus hajussüsteemides

Pelle Jakovits, jakovits@ut.ee

Kevad 2025

Loengu sisukord

- Hajussüsteemide komponentide suhtlus
  - Protsessidevaheline side
- Andmete edastamise viisid
- MQTT
- AMQP

Hajussüsteemide komponentide suhtlus

- Kuidas on hajusad komponendid on omavahel ühendatud?
- Kuidas toimub komponentide vaheline andmevahetus ja sünkroniseerimine?
- Kuidas need komponendid ja nende vahelised liidesed on konfigureeritud ühiseks (hajus)süsteemiks?
- Kas (asendatavad) komponendid on selgelt määratletud liidestega?

Definitstioon: Protsessidevaheline side

- Inglise keeles: Inter-process communication (IPC)
- Tüübid
  - Läbi jagatud mälu
  - Otseühendus **sokklite** kaudu
  - Läbi **teadete edastuse**
  - Läbi kaugprotseduuride ja hajusobjektide
  - Muud:
    - Torud (ühe suunalised), Failid

SOKKLID

Definitstioon: Sokkel

- Sokkel on side otspunkt
- Sokli aadresskoosneb IP aadressist ja pordist:
  - IP-aadress: 172.17.68.56
  - pordi number: 8080
  - Lisaks on tähtis protokoll: TCP, UDP
- Näited:

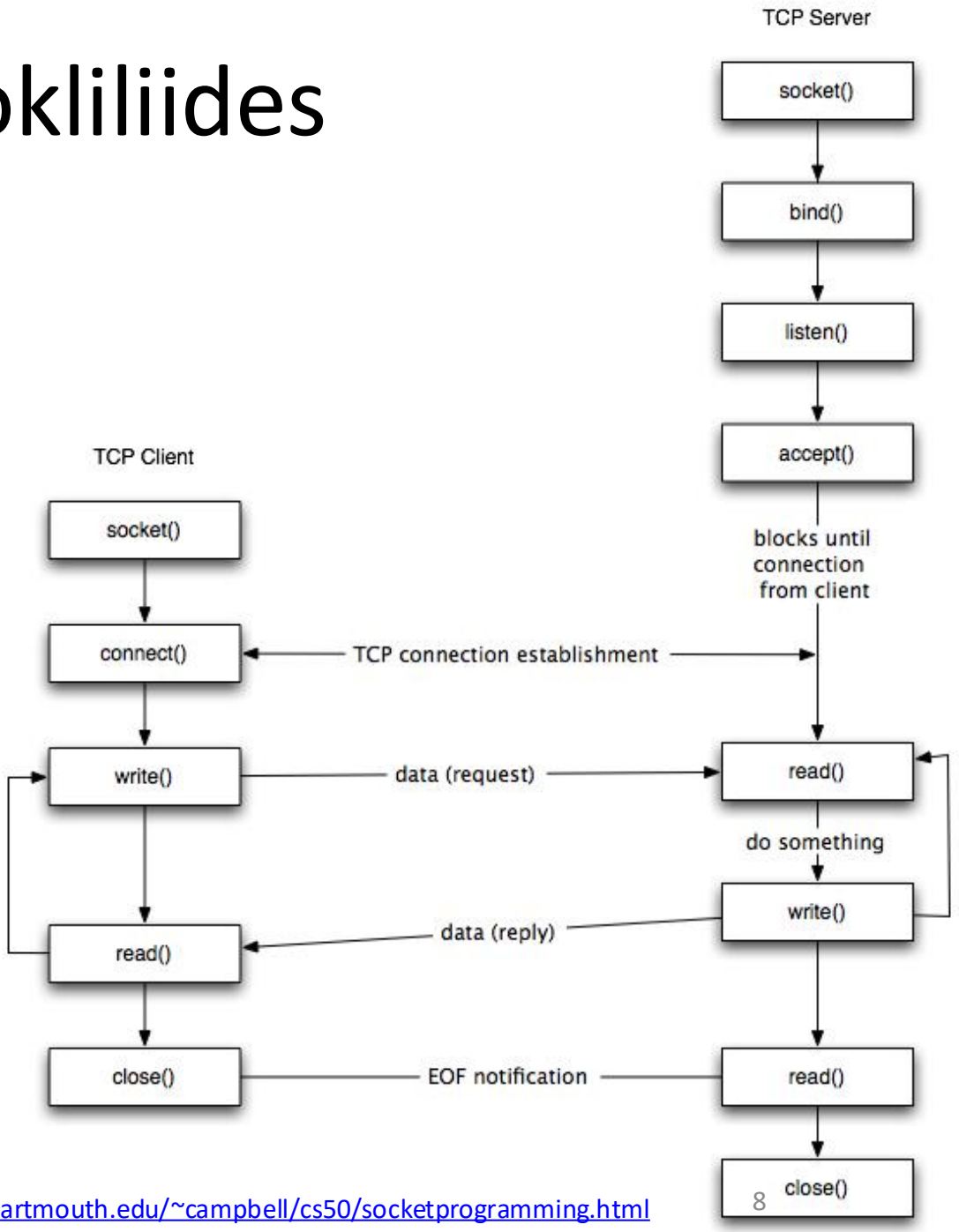
- Linuksis on Sokkel spetsiaalset tüüpi failipide
  - Kirjutame binaarsed andmed sokkli failipide kaudu
  - Loeme binaarseid andmeid sokkli failipidest

Sokliühendused

- Võimaldab protsessidel omavahe suhelda – sarnaselt nagu andmeid kirjutatakse või loetakse failidest
- Loome side kahe sokkli vahel, samas või erinevas arvutis
- Standardiseeritud liides (BSD socket API)
- Pakettside: UDP, SCTP, DCCP
- Voogside: TCP
- Saatjal ja vastuvõtjal on kohalikud puhvrid
- Nõuab madala taseme programmeerimist

Sokliliides

- **socket** — sokli loomine
- **bind** — sokli sidumine kohaliku aadressiga (Liides ja port)
- **listen** — ühenduste vastuvõtuks ressursi eraldamine
- **accept** — uue ühenduse ootamine ja vastuvõtt
- **connect** — teise osapoolega ühenduse algatamine
- **send** — soklisse andmete saatmine
- **receive** — soklist andmete lugemine
- **close** — sokli sulgemine



Allikas: https://www.cs.dartmouth.edu/~campbell/cs50/socketprogramming.html

TEADETE EDASTUS

Teadete edastus

- Sõnumite vahetamine protsesside vahel
- Olukorras kus jagatud mälu ei ole kasutusel
- Sokklipõhine ühendus liiga madalatasemeline
- Tüübid:
  - Sõnumi edastuse abstraktsioon (Message passing)
    - **MPI**
    - RPC
    - SOAP, REST
  - Sõnumite järjekorrad (Message queue)
    - **AMQP**
    - **MQTT**

Sõnumite-põhine ühendus

- Inglise keeles: Message-oriented
- Kaks peamist tüüpi
  - Sõnumite edastamine otse protsesside vahel
    - Võib olla nii sünkroonne kui asünkroonne
  - Sõnumite edastamine postkastide/järjekordade kaudu
    - Eesmärk on kõrgema-taseme püsiv asünkroonne suhtlus:
    - Protsessid saadavad üksteisele sõnumeid, mis pannakse järjekorda
    - Saatja ei pea ootama kohest vastust, vaid võib teha muid asju
    - Sõnumite järjekordade vahevara vastutab sageli tõrketaluvuse eest, et sõnumid kaduma ei läheks

MPI

- Message Passing Interface (MPI)
- Kõrgema abstraktsioonitaseme teadeteedastus-API
- Tootjast sõltumatu (tootjaspetsiifilised madalamad kihid)
- Paralleelrakendustele grupi sees suhtlemiseks
- Veakäsitluse eeldus: retry pole väga pikalt vajalik, kui läheb tuksi, on kõik tuksis

Sõnum / Teade

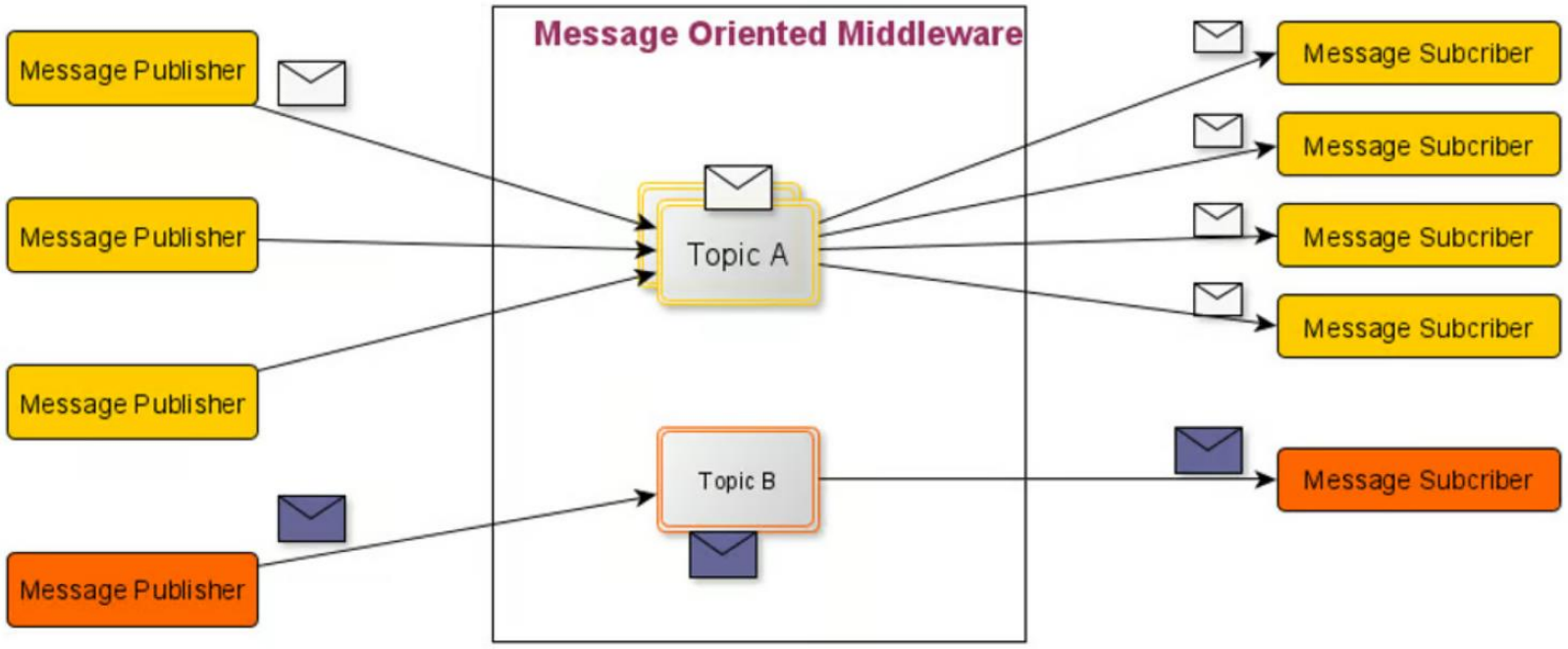
- Võib olla ükeskõik, mis:
  - Int väärtus
  - Int Array
  - JSON objekt
  - XML objekt
- Klient ja Server epavad kokku leppima, mis andmetüüpi saadetakse
- Madalamal tasemel sadetakse objektna

MPI primitiivid

- MPI_Send — saatmine, oodates kopeerimist kohalikku või kaugpuhvrisse
- MPI_Recv — teate lugemine, vajadusel oodates
- MPI_Sendrecv — saatmine vastuse ootamisega
- MPI_Isend — Mitteblokeeriv sõnumi saatmine saadetava teate viida edasiandmine
- MPI_Irecv — Mitteblokeeriv sõnumi vastuvõtmine
- MPI_Wait — konkreetse asünkroonse teate taga ootamine
- MPI_Waitany — hulgast suvalise asünkroonse teate taga ootamine
- MPI_Test — konkreetse pideme valmisoleku kontroll
- Kollektiivsed operatioonid paralleelarvutuste jaoks:
  - MPI_Bcast – Saadame sõnumi kõigile protsessidele
  - MPI_Gather – kogume sõnumi(d) kõigilt kokku ühele
  - MPI_Scatter – Jagame sõnumite listi kõigi osalejate vahel

Teatejärjekordadega suhtlus

- Teatejärjekord annab võimaluse, et saatja ja vastuvõtja pole samal ajal aktiivsed
- Vahendajaks on teateid mõnda aega säilitav järjekord
- Järjekordi võib mitu olla
- Lisaks on igal saatjal ja vastuvõtjal oma kohalik järjekord
- Kohalejõudmise garantiid ei ole, vahevara toimetab millagi teate kohale ja kas/millal seda loetakse, on vastuvõtja asi



Allikas: https://dev.to/tranthanhdeveloper/point-to-point-and-publish-subscribe-messaging-model-41j0

Teatejärjekordade primitiivid

- **put** — teate järjekorda lisamine
- **get** — blokeerub kuni järjekorras on mõni teade, tagastab esimese teate
- **poll** — tagastab esimese teate kui sellist on, ei blokeeru
- **notify** — seab tegevuse teadete tuleku peale automaatselt käivitamiseks
  - Notify jaoks erinevaid tehnilisi mehhanisme, sh "äratamine"

Süsteemi arhitektuur

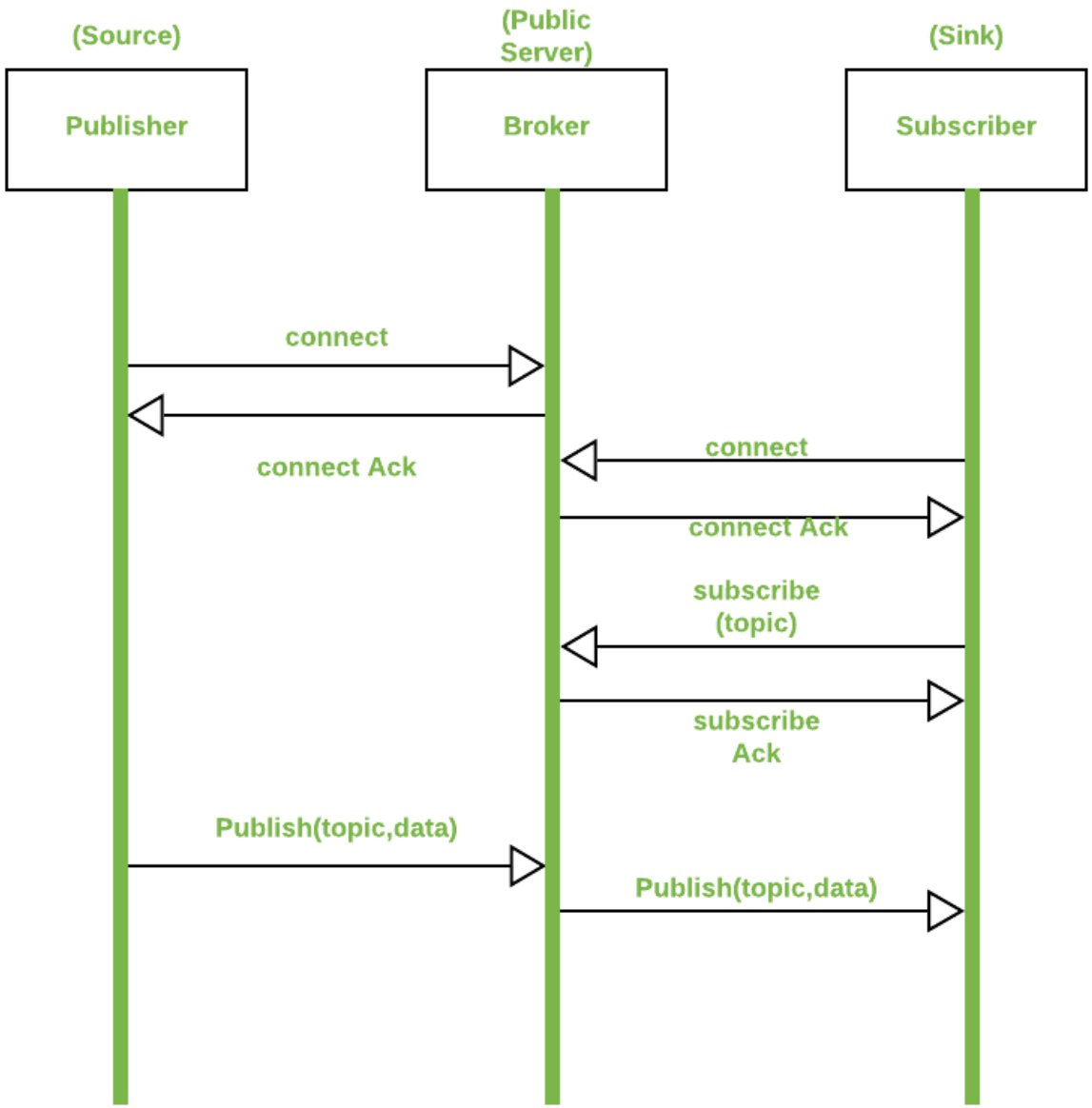
- Lähtejärjekord
- Sihtjärjekord
- Järjekordade nimed, nende asukohtade andmebaas
- Järjekorra haldurid
- Releed (vahendajad)
- Tekib järjekordade võrk tegeliku võrgu peale uueks kihiks
  - Marsruutimise probleemid
  - Ruutingutabelid
  - Marsruutimise haldamine valdavalt käsitsi

Järjekordade maaklerid, haldurid (brokers)

- Teate formaatidega on põhimõtteline probleem — teated võivad sisaldada mida iganes
- Formaati ette kokku leppida on enamasti võimatu ⇒ lepime ja arvestame eri formaadis teadetega
- Spetsiaalsed sõlmed tõlgivad neid läbivaid sõnumeid
- Süsteemi arhitektuuri mõistes on need tavalised rakendused järjekordade otsas

- Avaldamise-tellimise (publish-subscribe) mudeli võrguprotokoll, mis edastab sõnumeid seadmete vahel
- Üldjuhul seatakse üles eraldi sõlmena/serverina
- Selle kaudu vahetatakse andmeid hajussüsteemide komponentide, sõlmede vahel
- Sobib hästi asünkroonseks suhtluseks ning süsteemi skaleerimiseks
  - N sõnumite tootjat
  - M sõnumite kuulajat
  - Sõnumid jagatakse M kuulaja vahel – ei vaja keerulist sünkroniseerimist

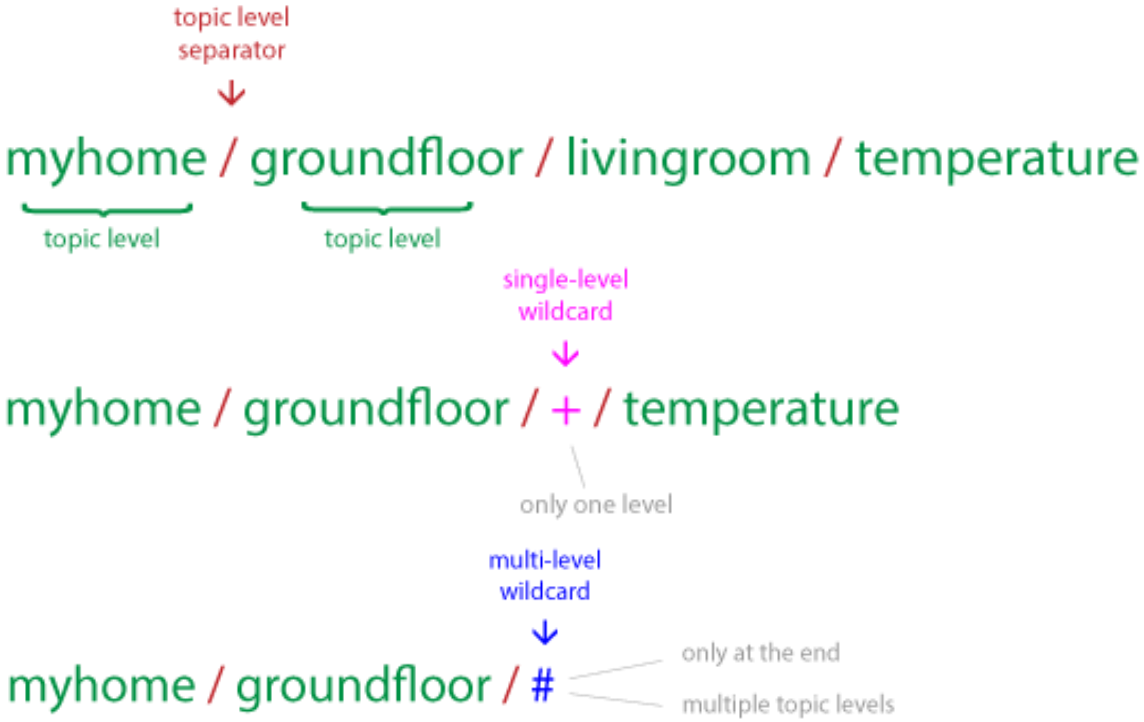
Lihtsustatud illustratsioon kuidas MQTT töötab:



Broker Based MQTT Protocol

MQTT

- Reaalselt ei ole eraldi järjekordi!
  - Teemade põhine marsruutimine

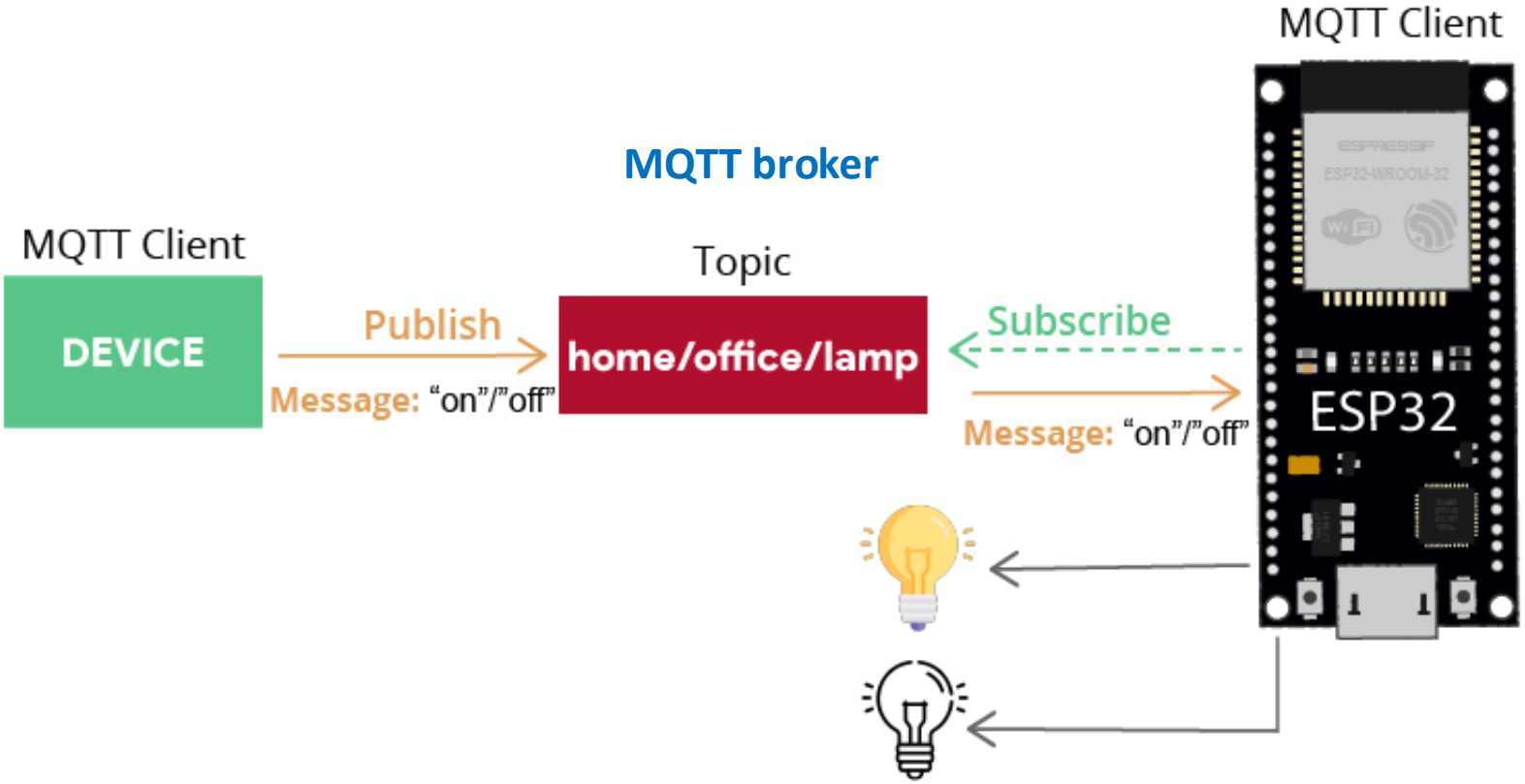


https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/

MQTT teenuse kvaliteet

- Teenuse kvaliteet – Quality of Service (QoS)
  - **Kõige rohkem üks kord** (0) – sõnum edastatakse kõige rohkem üks kord või ei edastata seda üldse
  - **Vähemalt üks kord** (1) – sõnum edastatakse alati vähemalt ühe korra
  - **Täpselt üks kord** (2) – sõnum edastatakse alati täpselt üks kord

MQTT näide



https://randomnerdtutorials.com/what-is-mqtt-and-how-it-works/

AMQP

- Algatatud 2006, V1.0 2011 aastal.
- Advancd Message Queing Protocol
- Avatud standard rakenduskihi protokoll sõnumitele orienteeritud vahevara jaoks
- Asünkroone, järjekordade põhine
- Sõnumid jäävad järjekorda, kuni neid sealt võetakse
- SASL-il või TLS-il põhinev autentimine ja krüptimine
- Implementatsioonid:
  - RabbitMQ, Apache ActiveMQ, Azure Service Bus
  - Amazon MQ (ActiveMQ, RabbitMQ hallatud teenus)

AMQP olemid

- **Exchange** – **Ühendussõlm**, kuhu saab sõnumeid saata
  - Ajutine või pidev
- **Queue** - **järjekord**, kust saab sõnumeid kuulata, küsida
- **Routing Key** - **Võti**, mis kirjeldab sõnumi tüüpi, ja mida kasutatakse sõnumite filtreerimiseks ja marsruutimiseks
  - Näiteks: "myfloor.livingroom.temperature"
- **Binding – Seos,** mis määrab millise võtmega sõnumid suunata ühendusõlmest X järjekorda Y.
- **Message** - **Sõnum** mida saadetakse
  - Header: Võti-Väärtus stiilis metainfo, näiteks **routing key**
  - Body: binaarne objekt

AMQP ja MQTT erinevused

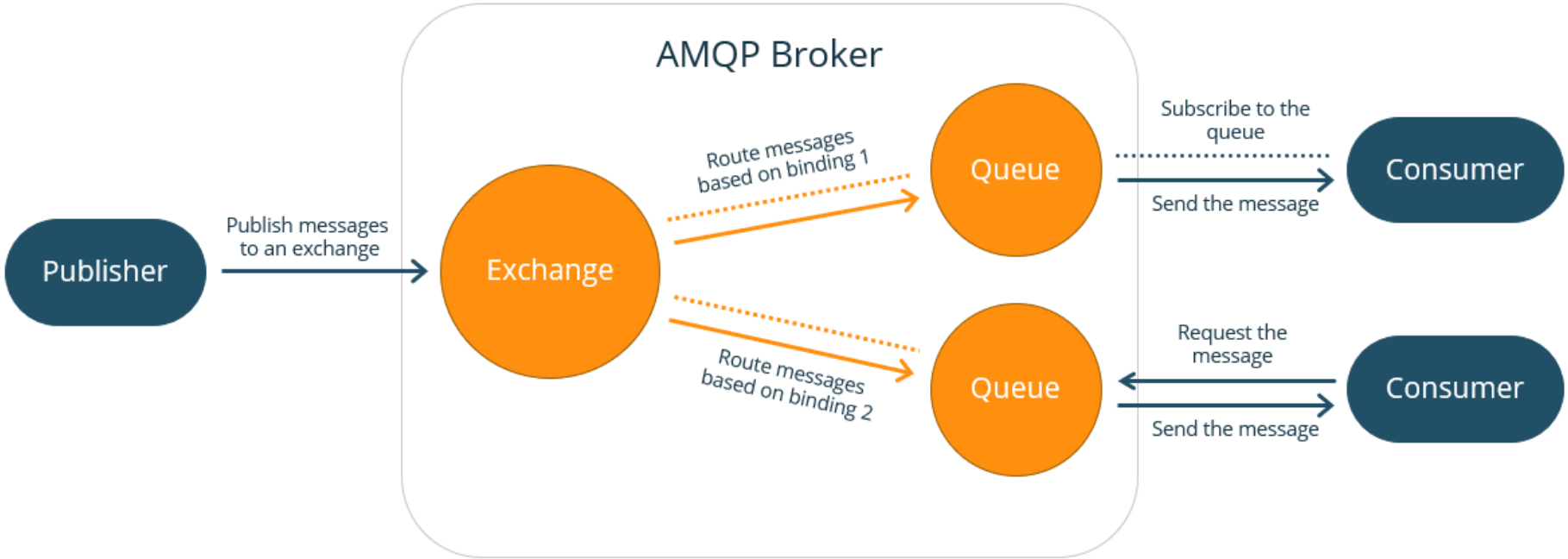
- Ühendussõlmed - Andmeid ei saadeta otse järjekorda, vaid ühendussõlme
- Ühendussõlmede ning järjekordade vahel toimub marsruutimine
- Järjekorrad ei ole samad, mis MQTT teemad, vaid pigem postkastid, kuhu teatud teemaga sõnumid kohale jõuavad

AMQP liidese operatsioonid

- AMQP Declare **Exchange** – loob **ühendussõlme**
- AMQP Declare **Queue** - loob **järjekorra**
- AMQP **Bind Queue** - seob järjekorra ühendussõlmega
- AQMP **Publish** - saadab sõnumi ühendussõlme
- AMQP **Receive** - võtab järjekorrast sõnumi

AMQP

- Sõnumite avaldamise ja tarbimise muster
  - Publish-subscribe (consume)



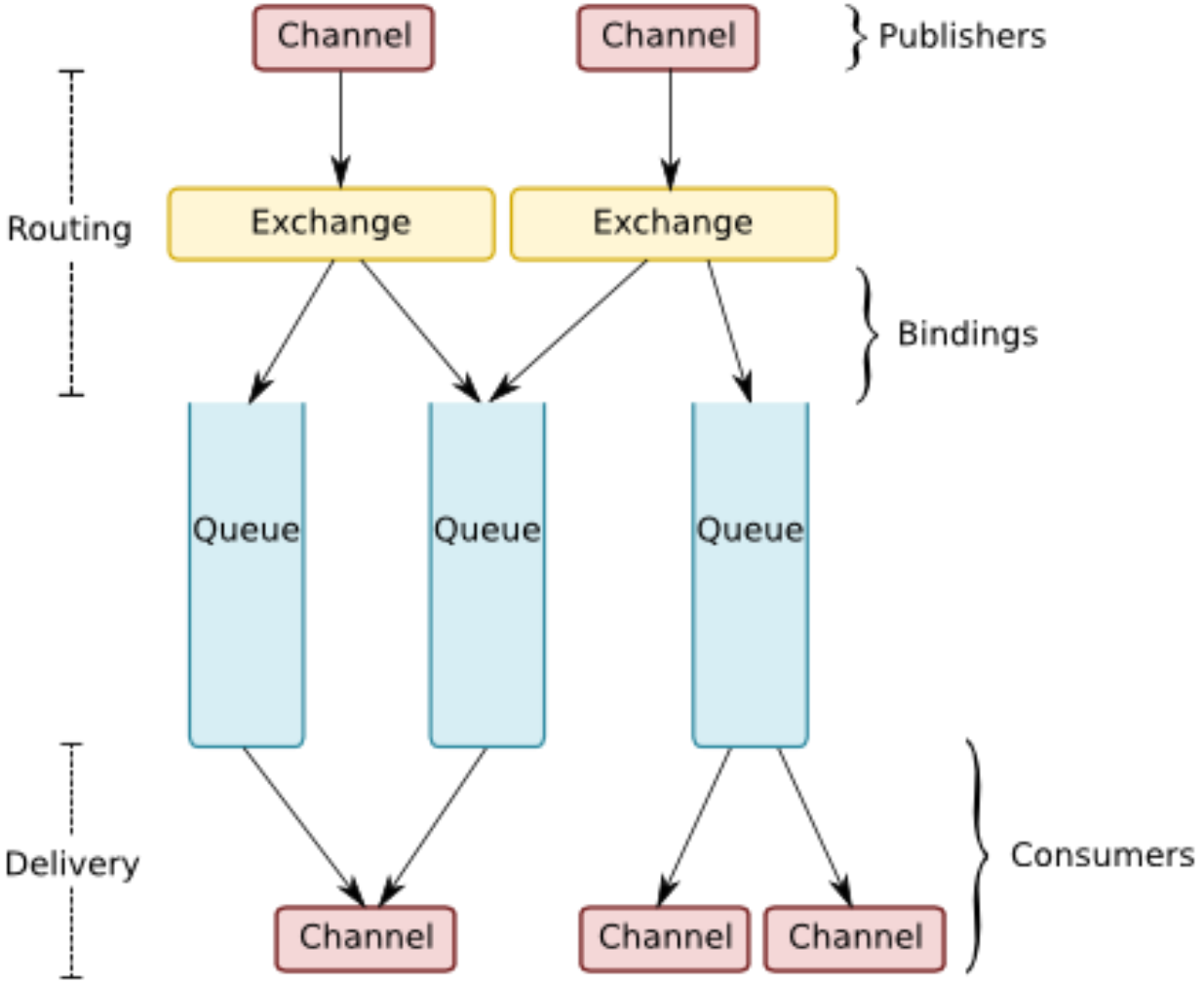
Sõnumite marsruutimine

- Nii seosed (binding) kui ka sõnumid sisaldavad **marsruutimisvõtit** (routing key), mida maakler kasutab sõnumi suunamiseks. Näited:
  - **tartu.delta.välisvalgustase**
  - **tartu.delta.korrus3.ruum3040.temp**
- Ühendussõlmede tüübid:
  - ***Direct*** - *Ühendussõlm* suunab sõnumid järjekordadesse, mille marsruutimisvõti on identne sõnumi võtmega.
  - ***Fanout*** - Saadab sõnumid kõikidesse seotud järjekordadesse. Sõnumi suunamise võtit ignoreeritakse.
  - ***Topic*** - suunab sõnumeid seotud järjekordadesse, kui sõnumi marsruutimisvõti ühtib sidumismarsruutimisvõtmes määratud mustriga.
    - Toetab metamärke * ja #.
      - * - üks sõna
      - # - mitu sõna
      - **tartu.delta.korrus3.*.temp**

RabbitMQ

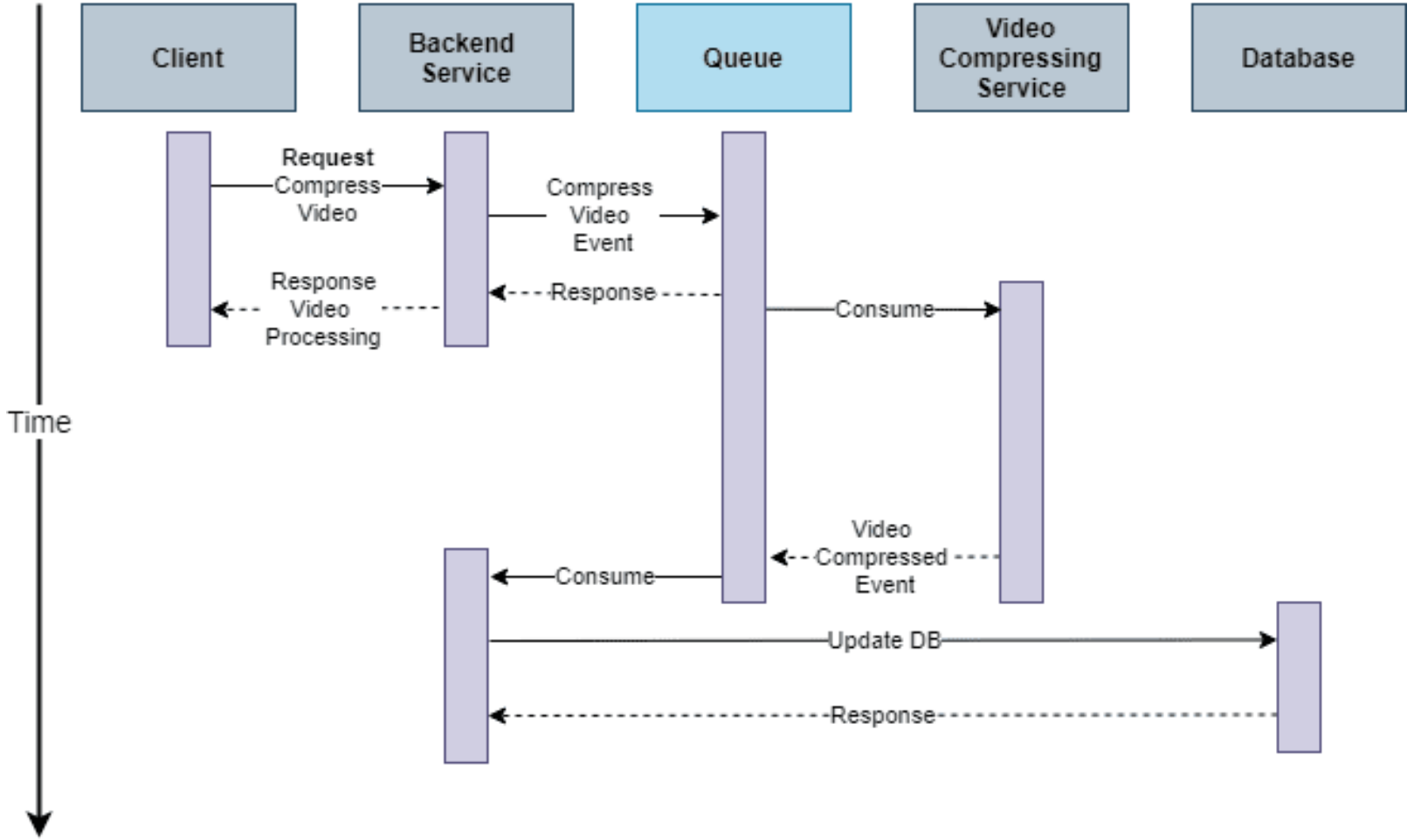
- AMQP protokolli implementatsioon
- Avatud lähtekoodiga (https://github.com/rabbitmq).
- Saadaval on palju pistikprogramme (mqtt, ajastatud kohaletoimetamine, e-posti teel saatmine, prioriteetsed järjekorrad)
- Lisa funktsioonid:
  - Klasterdamine
  - Veebihaldusliides
  - Kasutajataseme juurdepääsukontroll

AMQP



https://pythonhosted.org/nucleon.amqp/amqp.html

Sõnumite saatmise mustrid: Asünkroonne päring-vastus koos järjekorraga



Source: https://dev.to/ragrag/asynchronous-request-response-pattern-2pbj

Järgmine loeng

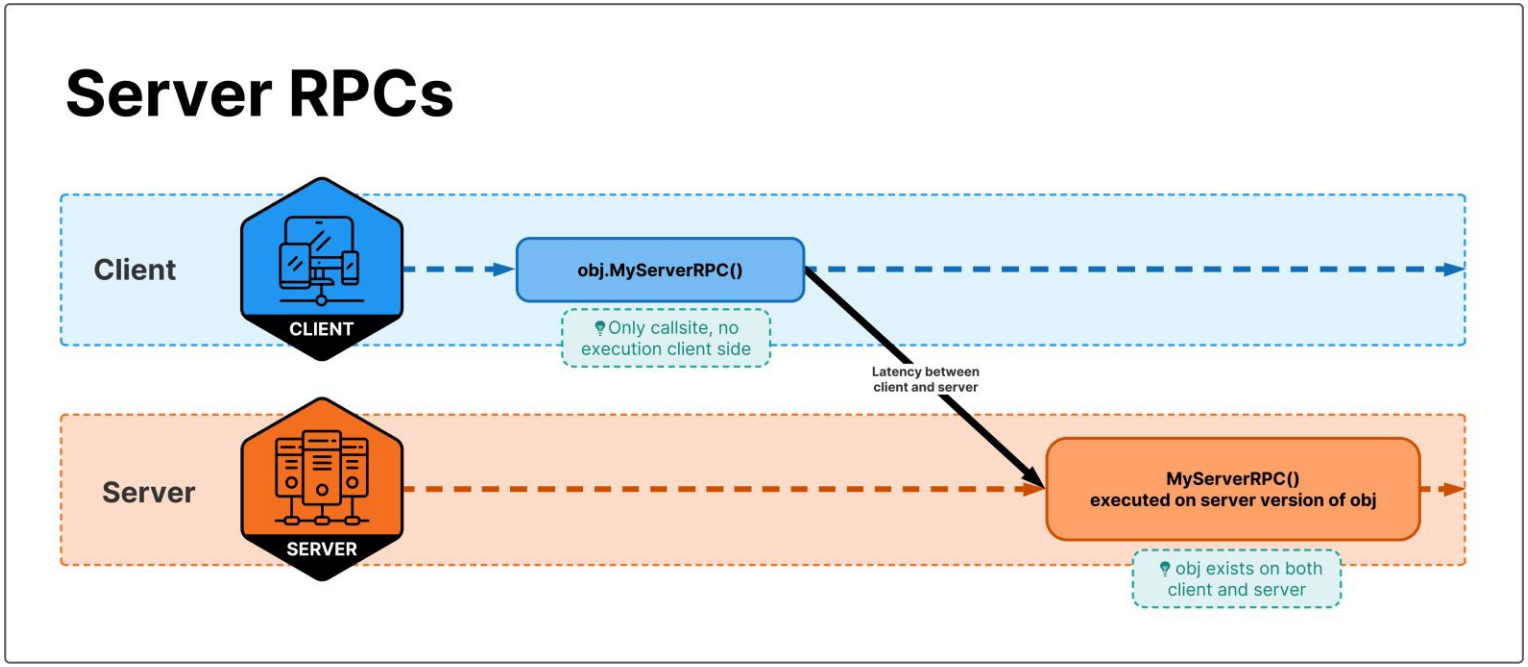
- Jätkame andmete vahenduse ja nende liideste teemal
  - Remote Procedure Call (RPC)
  - Hajusobjektid
- Ülejärgmisel loengul
  - REST
  - SOAP
  - Veebi API'd

Kaugprotseduurid

- Remote Procedure Call (RPC)
- Programmeerijad tunnevad lihtsat protseduurimudelit: Kutsume välja protseduuri/meetodi teises arvutis.
- Kohaliku protseduuri väljakutse laiendus protseduuri välja kutsumiseks ka ugarvutist
- Hästi kavandatud protseduurid toimivad isoleeritult (must kast).
- Helistaja ja kutsutava vahelise suhtluse saab peita, kasutades protseduuri-väljakutse mehhanismi
- Jäik klient-server mudel
- Sünkroonne suhtlus
- Vähene transparentsus

Kaugprotseduurid

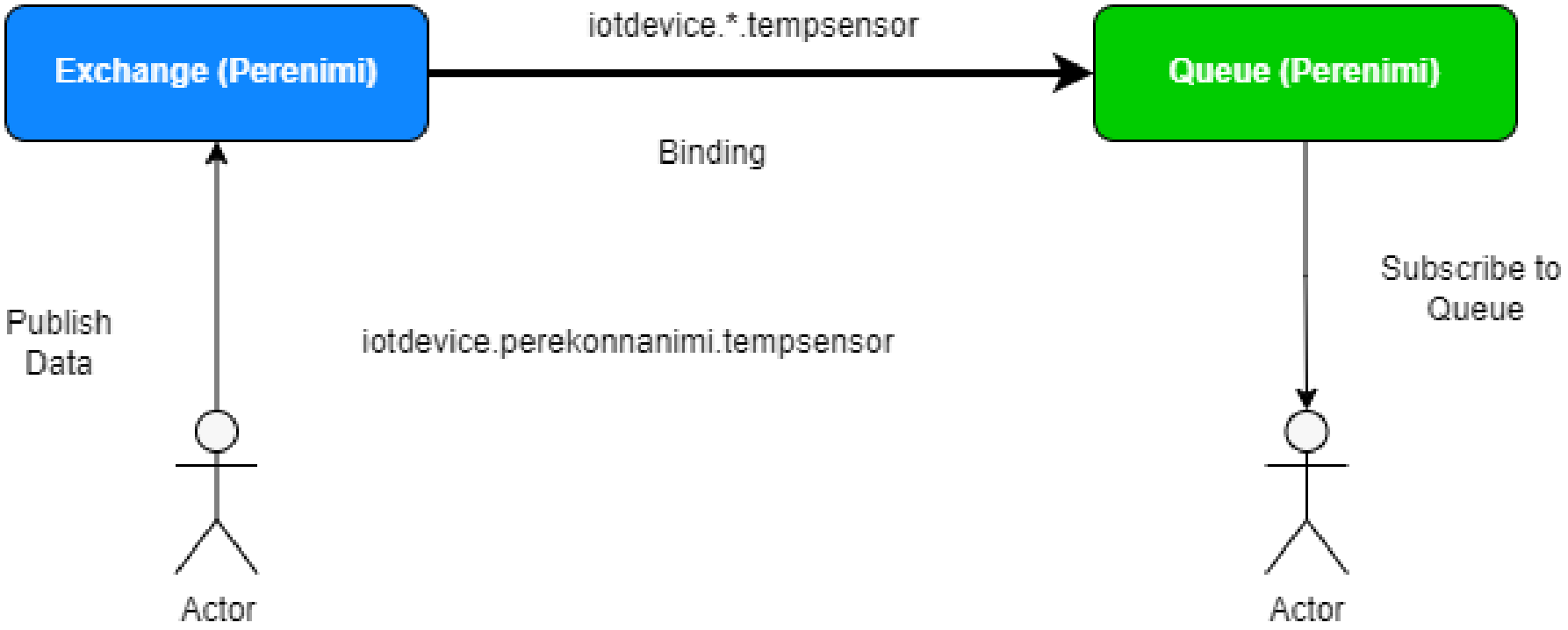
- Kliendil on kood protseduuri/meetodi välja kutsumiseks, aga implementatsiooni ei ole
- Meetodi väljakutsumise asemel, edastatakse sõnum meetodi väljakutsumiseks serverile
- Serveris on sama meetod koos implementatsiooniga
  - Lisaks ka viis tulemuse edastamiseks kliendile



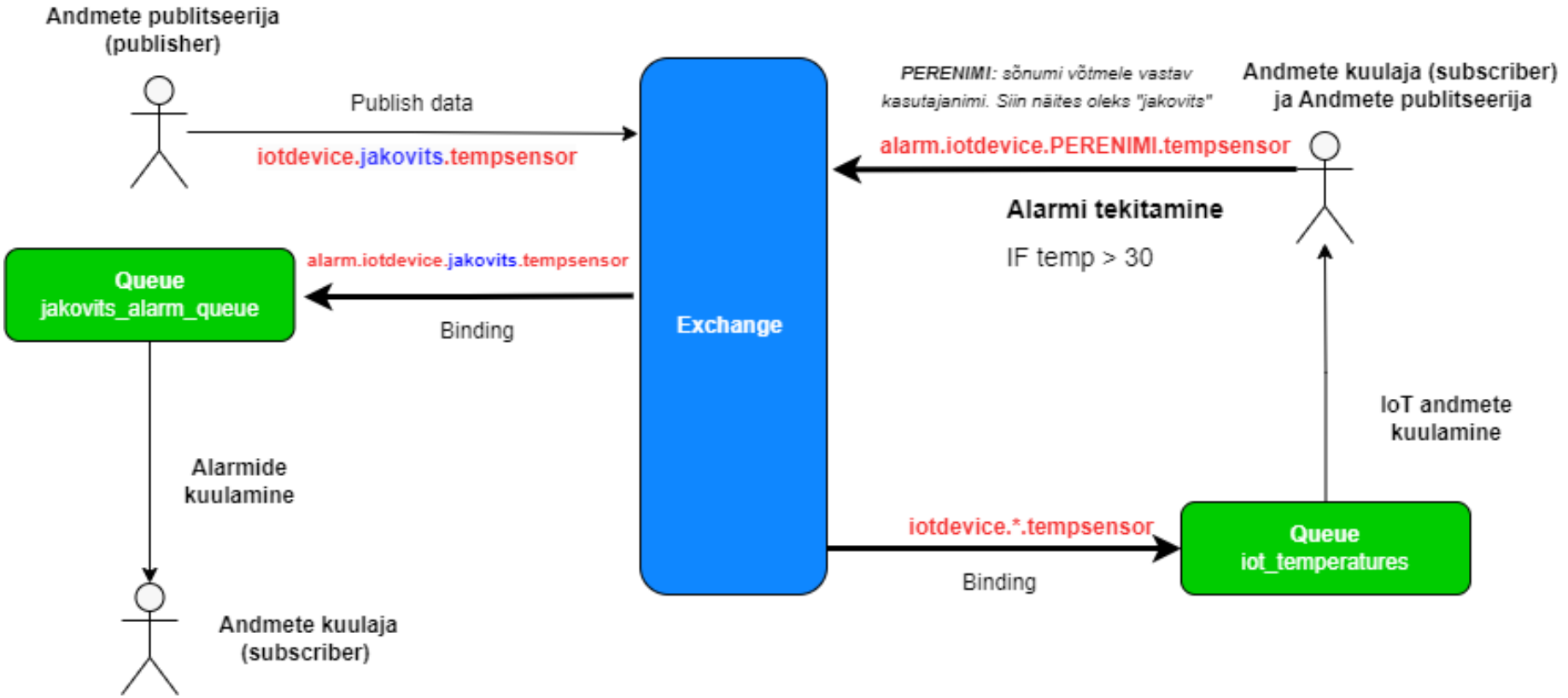
Selle nädala praktikum

- Sõnumite edastus hajusüssüsteemides Teate-järjekordade kaudu
  - Python hajusrakenduse loomine
  - RabbitMQ andmehaldur

Praktikumi ülesanded



Praktikumi ülesanded



Allikad ja viited

- Van Steen, Maarten, Tanenbaum, Andrew. Distributed Systems: Principles and Paradigms (Third edition). Published by Maarten van Steen, 2017.
  - Tasuta versioon: https://www.distributed-systems.net/
- Hajussüsteemide aine materjalid, Meelis Roos, Tartu Ülikool