Veebiteenuste ja hajussüsteemide arendus

Loeng 4: Kaugprotseduurid

Pelle Jakovits, jakovits@ut.ee

Kevad 2025

Suhtlusviisid

- Sokliühendused
- Sõnumi edastus (Message passing)
  - MPI
  - **RPC,**
  - SOAP, REST
- Sõnumite järjekorrad (Message queues)
  - MQTT, AMQP
- Muud:
  - Torud (ühe suunalised), Jagatud mälu, Failid

Idee

- Sõnumite saatmise asemel:
  - Kas protsess saaks välja kutsuda teise protsessi meetodi?
    - Samas või teises serveris asuv protsess
  - Sarnaselt programmeerimises teise klassi objekti meetodite välja kutsumisele.

Kaugprotseduurid

- Remote Procedure Call (RPC)
- Idee aastast 1984 (Birell ja Nelson)
- Kohaliku protseduuri/funktsiooni väljakutse laiendus protsedu uri/funktsioonide väljakutsumiseks kaugarvutist
  - Ei ole ühtegi põhjust, miks funktsioone ei saaks välja kutsuda üle võrgu
- Klient-Server mudel
- Protseduurid toimivad musta kastina
  - Klient ei tea kuidas implementeeritakse
  - Aga on vaja teada, mis operatioon välja kutsutakse
    - Mis on operatsiooni sisendid

Kaugprotseduuride defineerimine

- Kaugprotseduuride kirjeldus defineerib sisend- ja väljundparameetrid
- Kaugprotseduuri töökeskkond on protseduuri väljakutsuja keskkonnast erinev
  - Operatsioonisüsteem, aadressruum, jne.
- Parameetrid ja tulemus edastatakse keskkondade vahel sõnumitena
- Ühendus suhteliselt kitsaste kanalite kaudu
- Sünkroonne täitmisjärje üle andmine
  - Klient jääb ootele kuni operatsiooni täidetakse

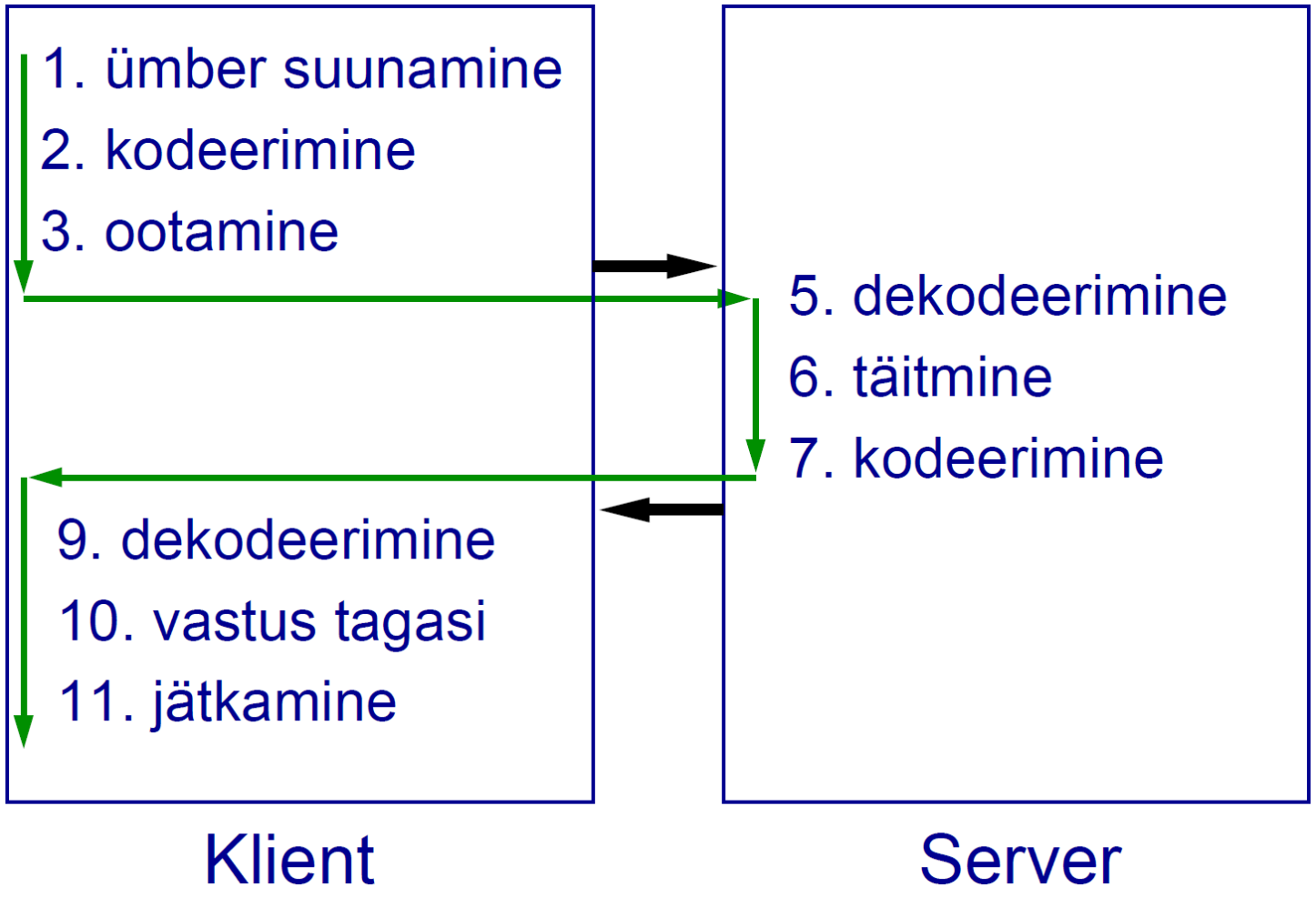
Kaugprotseduurid

- 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 tagastamiseks kliendile
- Kliendi ja serveri vaheline suhtluse peidetakse ära protseduuri väljakutsumise mehhanismi kaudu

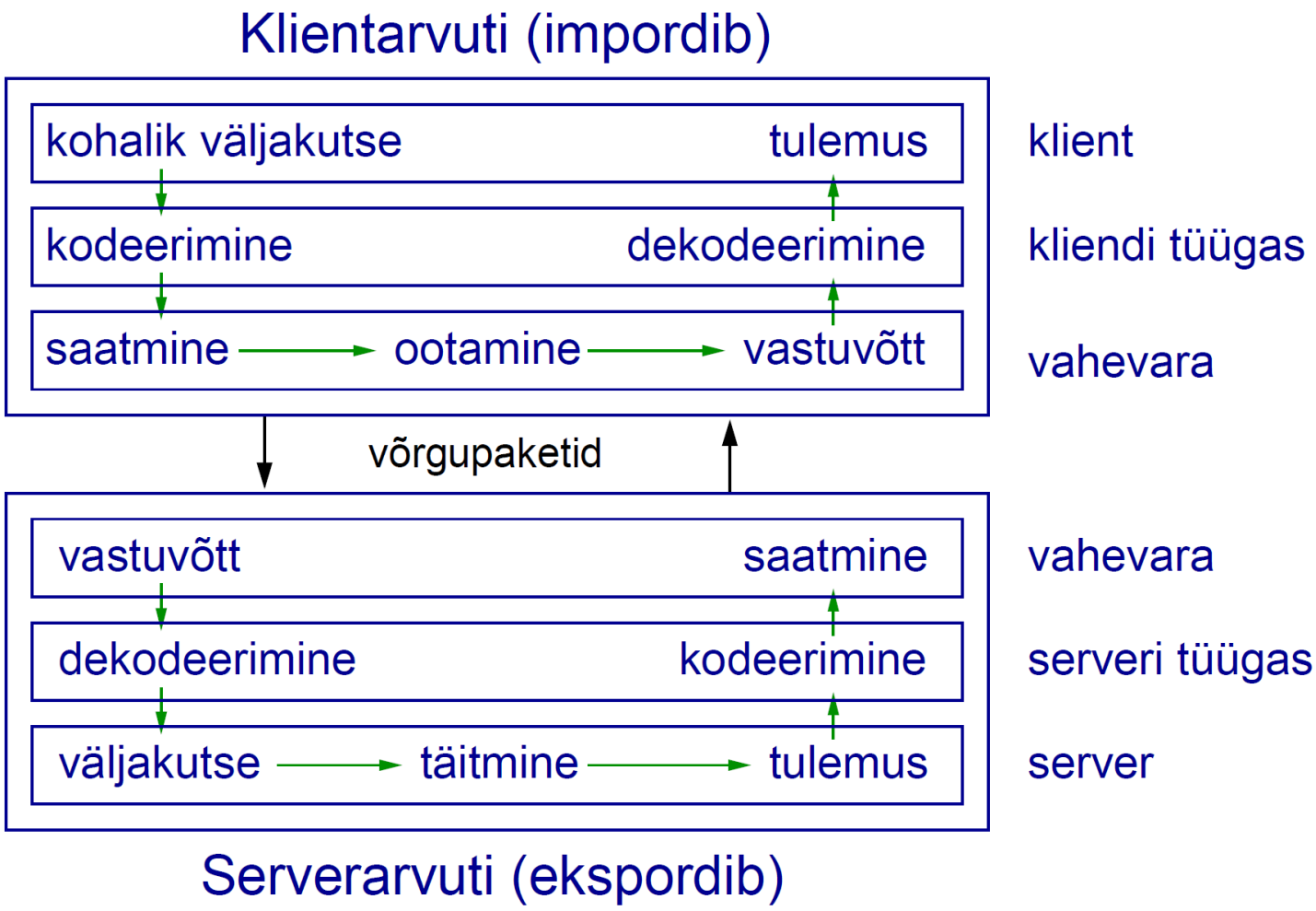
Tüügas - ik. Stub

- Hajussüsteemides on tüügas/stub programm või objekt, mis toimib kaugteenuse või objekti ajutise asendusena.
- Võimaldab klientrakendusel pääseda teenusele ligi nii, nagu see oleks kohalik
  - varjates aluseks oleva võrgusuhtluse üksikasju
- Lihtsustab arendusprotsessi
  - klientrakendus ei pea olema teadlik hajutatud andmetöötluse keerukusest.
- Selle asemel võib see kaugsuhtluse haldamisel tugineda tüükale, pakkudes samal ajal arendajale tuttavat liidest

RPC käsu täitmine



RPC süsteemi kihid



Tõrgete käsitlemine

- RPC kasutamine võib mitte õnnestuda, sellest tuleb klienti teavitada
  - Vead tegelikus protseduuris
  - Sidevõrgu tõrked
- Vigade käsitlemine võib olla realiseeritud programmeerimiskeele tasemel
- Keelest sõltumatu RPC mehhanism peab omama ka vigadest teatamise viisi

Edastuse kordamine

- Sidetõrke puhul saame teha valikuid kolmes asjas:
  - Kas päringut korrata kliendi poolt?
  - Kui saata päringut mitu korda, kas siis filtreerida serveri pool duplikaate?
  - Kas korrata vastuseid serveri poolt?

Veahalduse semantika

- **Võibolla** — ühekordne täitmise üritamine ilma veakontrollita, "ebaoluliste" asjade jaoks
- **Vähemalt üks kord (at least once)**
  - Protseduuri tuleb vähemalt kord täita
  - Kordamine ilma duplikaatide filtreerimiseta
- **Ülimalt üks kord (at most once)**
  - Duplikaatide filtreerimine ja vastuste kordamine
  - Kõige sagedamini kasutusel
- **Täpselt üks kord (exactly once)**
  - Transaktsiooni põhimõte
  - Kindlasti jõuab kohale. Ei teki duplikaate

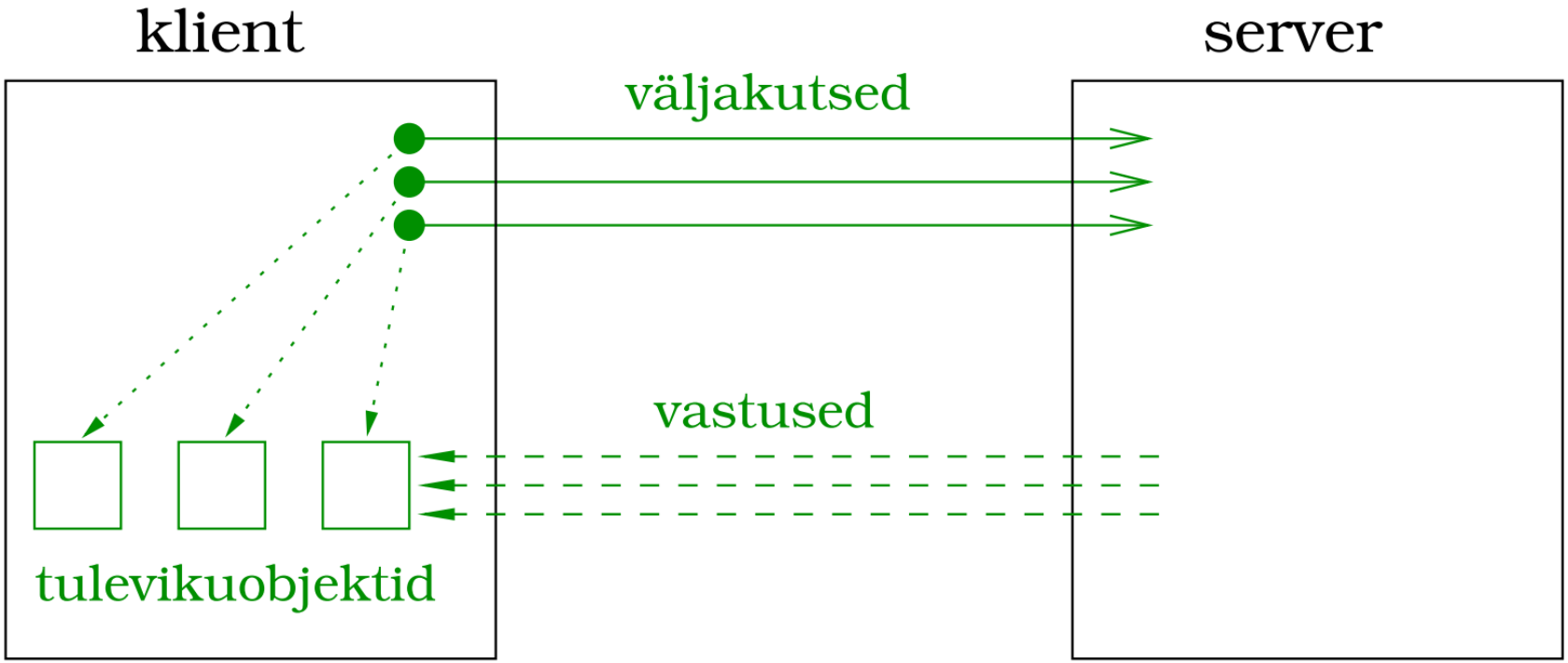
Kaugprotseduuride probleemid

- Sünkroonsus
- Väikesed ülekandeühikud
- Optimeeritud vastuse saamise ajale, mitte läbilaskevõimele
- Puuduvad vookontroll ja puhverdamine
- Vähene transparentsus (tüübid, globaalsed muutujad, . . .)
- Väljakutse ahelad rohkem kui kahe osapoolega süsteemides
- Jäik klient-server roll
  - Pole võrdsed suhtluspartnerid
  - Puuduvad vahetulemuste edastamise võimalus ja callback'id

Asünkroonne RPC

- Üheks liideseks asünkroonsuse saavutamiseks on tulevikuobjetid:
  - Asünkroonsel väljakutsel tagastatakse tulevikuobjekt (Future)
  - Tulemused tekivad transparentselt tulevikuobjekti sisse
  - Klient saab tulevikuobjekte vastuse suhtes testida ja kui objektis on vastus kohal, siis vastuse lugeda
  - Tulevikuobjekt tagastatakse koheselt päringu saatmisel ning klientprogramm saab oma täitmist jätkata
  - Range tüübikontroll tulevikuobjektide üle
- Liideseks võib olla ka lihtne madala taseme pollitav-oodatav liides
- Summaarselt kiirendab tööd juhul, kui võrgulatents on märgatav

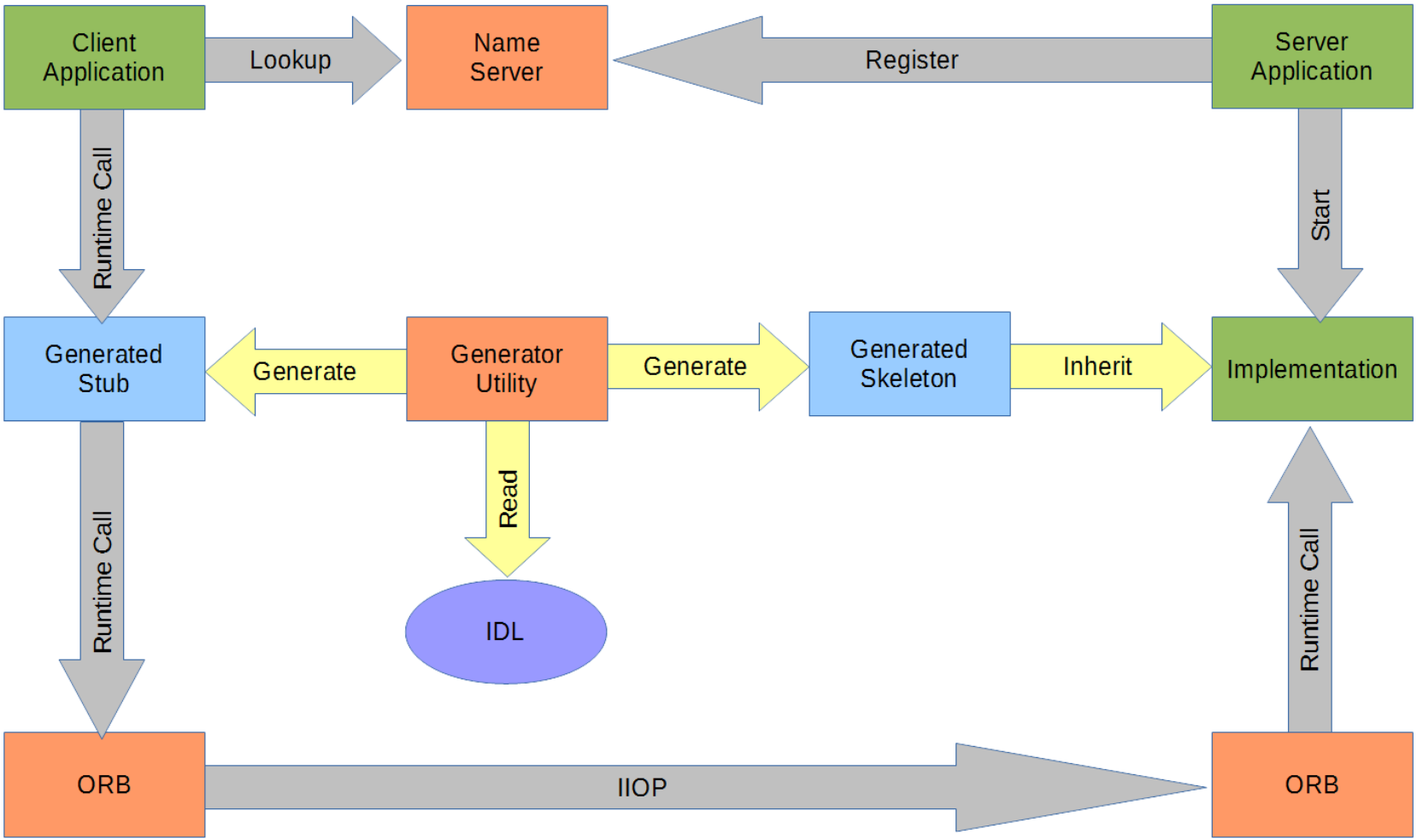
Asünkroonne RPC



Kaugprotseduuride liideste defineerimise standardid/keeled

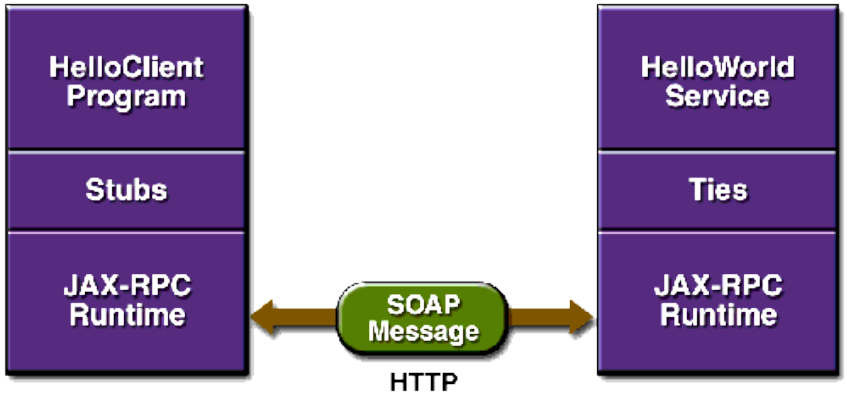
- **ONC IDL** - Open Network Computing interface definition language
- **CORBA IDL** – Common Object Request Broker Architecture interface definition language
- **MIDL** - Microsoft Interface Definition Language
- **JAX-RPC** - Java API for XML Based RPC
- **XML-RPC**
- **JSON-RPC**
- **REST – OpenAPI standard**

RPC teenuste otsimise ja implementeerimise voog COBRA näitel



JAX-RPC

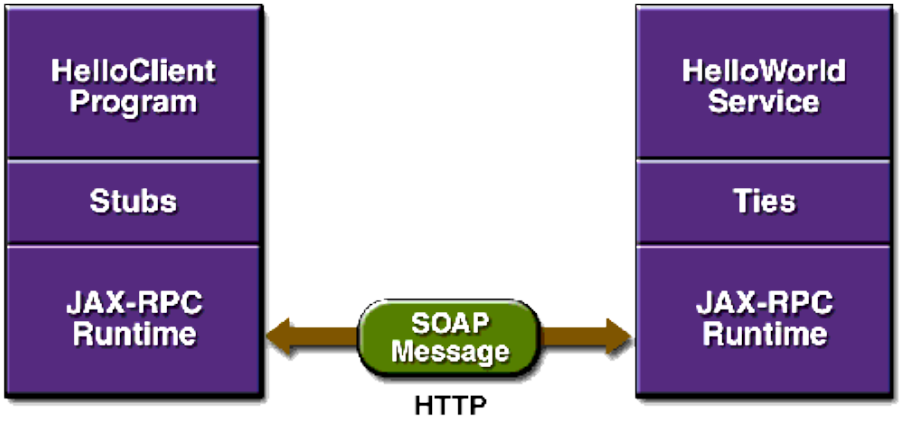
- Jakarta XML RPC (JAX-RPC) – Java API XML-põhise RPC jaoks
- Sisemiselt kasutab sõnumite edastamiseks SOAP-i
- Mõeldud veebiteenuste jaoks
- **wscompile** – genereerib kliendi ja serveri tüükad (*ik. stubs*)
- **wsdeploy** – genereerib veebiteenustega seotud failid (nt WSDL) ja pakendab need.)



http://www.cs.uccs.edu/~cs526/jwsdp/docs/tutorial/doc/JAXRPC3.html

Implementing JAX-RPC

- `HelloIF.java` - Teenust defineeriv liides
- `HelloImpl.java` - implementeerib `HelloIF` liidese
- `HelloClient.java` - klient, kes võtab teenusega ühendust ja kutsub seejärel välja sayHello meetodi
- **`config.xml`** - konfiguratsioonifail, mis määrab serveri asukoha
- **`jaxrpc-ri.xml`** - kirjeldab veebiteenust ja selle parameetreid
- **`web.xml`** - veebikomponendi – servleti – juurutamise kirjeldus, mille põhjal käivitatakse teenus



http://www.cs.uccs.edu/~cs526/jwsdp/docs/tutorial/doc/JAXRPC3.html

Java Liides (Interface)

```
import java.rmi.Remote;
import java.rmi.RemoteException;

public interface HelloIF extends Remote {

    public String sayHello(String s)
                            throws RemoteException;

}
```

http://www.cs.uccs.edu/~cs526/jwsdp/docs/tutorial/doc/JAXRPC3.html

Java Server

```
public class HelloImpl implements HelloIF {

    public String message = "Hello ";

    public String sayHello(String s) {
        return message + s;
    }
}
```

http://www.cs.uccs.edu/~cs526/jwsdp/docs/tutorial/doc/JAXRPC3.html

Java Klient

```
import javax.xml.rpc.Stub;

public class HelloClient {
    public static void main(String[] args) {
        try {
            Stub stub = createProxy();
            HelloIF hello = (HelloIF)stub;
            System.out.println(hello.sayHello("Duke!"));
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

   private static Stub createProxy() {
        // Note: MyHello_Impl is implementation-specific
        // MyHello_Impl class is generated automatically
        return (Stub)(new MyHello_Impl().getHelloIFPort());
    }
}
```

http://www.cs.uccs.edu/~cs526/jwsdp/docs/tutorial/doc/JAXRPC3.html

XML-RPC

- Lihtne XML esitusel põhinev RPC mehhanism
- Peaaegu minimalistlik võrreldes SOAP-iga
- Kasutab tavalist HTTP protokolli XML kujul päringute ja vastuste edastamiseks
- Lihttüübid, massiivid, struktuurid, string, datetime, base64

```
<?xml version="1.0"?>
   <methodCall>
   <methodName>examples.getStateName</methodName>
   <params>
         <param>
               <value><i4>40</i4></value>
         </param>
   </params>
</methodCall>
```

XML-RPC vastus

```
<?xml version="1.0"?>
<methodResponse>
     <params>
          <param>
               <value><string>South Dakota</string></value>
          </param>
     </params>
</methodResponse>
```

JSON-RPC

- JSON — JavaScript Object Notation
  - Javascripti parser on brauseritel sees olemas, ei vaja lisaparsimist
- JSON-RPC — kergekaaluline RPC üle HTTP, kasutades JSON formaati
- application/json MIME tüüp
- Notification — ühesuunaline suhtlus (ükskõik kummas suunas)
- Nimelised ja positsioonilised parameetrid
- error-tüüpi objektid vigadest teatamiseks
- Masinloetav teenuse kirjeldus (samuti JSON kujul)

JSON-RPC näide

- Päring: `service.subtract([42, 23])`

```
{
  "jsonrpc": "2.0",
  "method": "subtract",
  "params": [42, 23],
  "id": 1
}
```

- Vastus

```
{
  "jsonrpc": "2.0",
  "result": 19,
  "id": 1
}
```

Vea näide

```
{
    "jsonrpc": "2.0",
    "method": "_Error",
    "params": {
        "error": {
            "code": 1,
            "message": "subtract method parameter is
                        missing second argument."
        }
    }
}
```

Veel üks näide

- Tagastab hinnangu, kui palju gaasi on Etherium tehingu lõpuleviimiseks vaja

```
{
    "jsonrpc": "2.0",
    "method": "eth_call",
    "params": [
        {
            "from": "0x99d3f3a83a68b3c3bb923d86387dc91340c53304",
            "to": "0x99d3f3a83a68b3c3bb923d86387dc91340c53304",
            "gas": "0x76c0",
            "gasPrice": "0x9184e72a000",
            "value": "0x9184e72a",
            "data": "0x7f74657374"
        },
        "latest"
    ],
    "id": 67
}
```

Response:

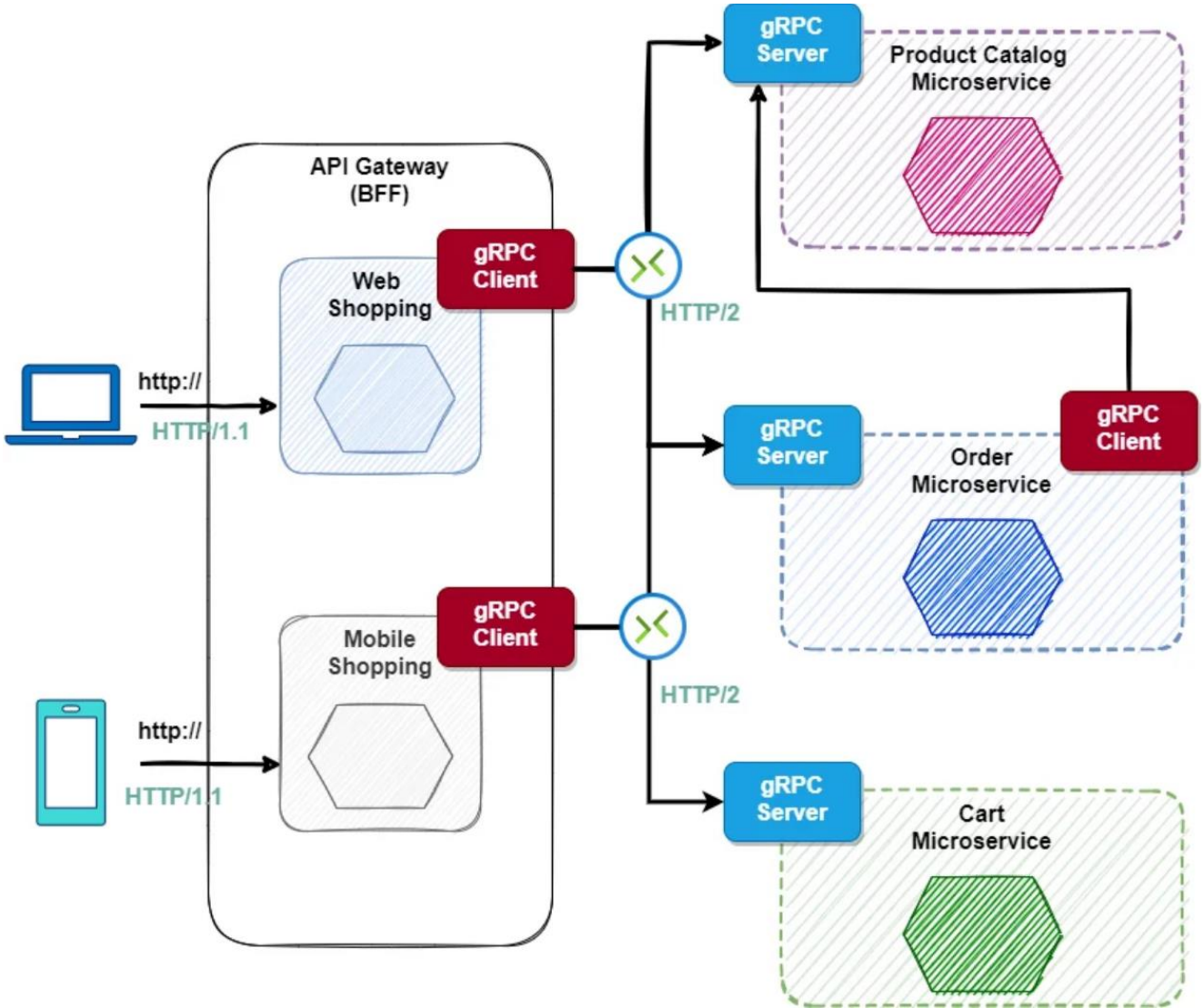
```
{
    "jsonrpc": "2.0",
    "id": 67,
    "result": "0x5208"
}
```

gRPC

- Cloud Native Computing Foundation (CNCF) incubation projekt
- Kasutuses mikroteenuste vahel HTTP päringute asendusena
- Kasutab HTTP/2 protokolli vanema HTTP/1.1 asemel
- Disainitud lahendama varasemate RPC protokollide puuduseid
- Kasutab Google protokolli puhvreid (Protocol Buffers) XML/JSON asemel
  - Kompaktnsm binaarne andmesalvestus
  - Kiire parsimine
  - Kasutatav paljudes programmeerimiskeeltes
  - Optimeeritud funktsionaalsus automaatselt genereeritud klasside kaudu

Allikas: https://techdozo.dev/grpc-for-microservices-communication/

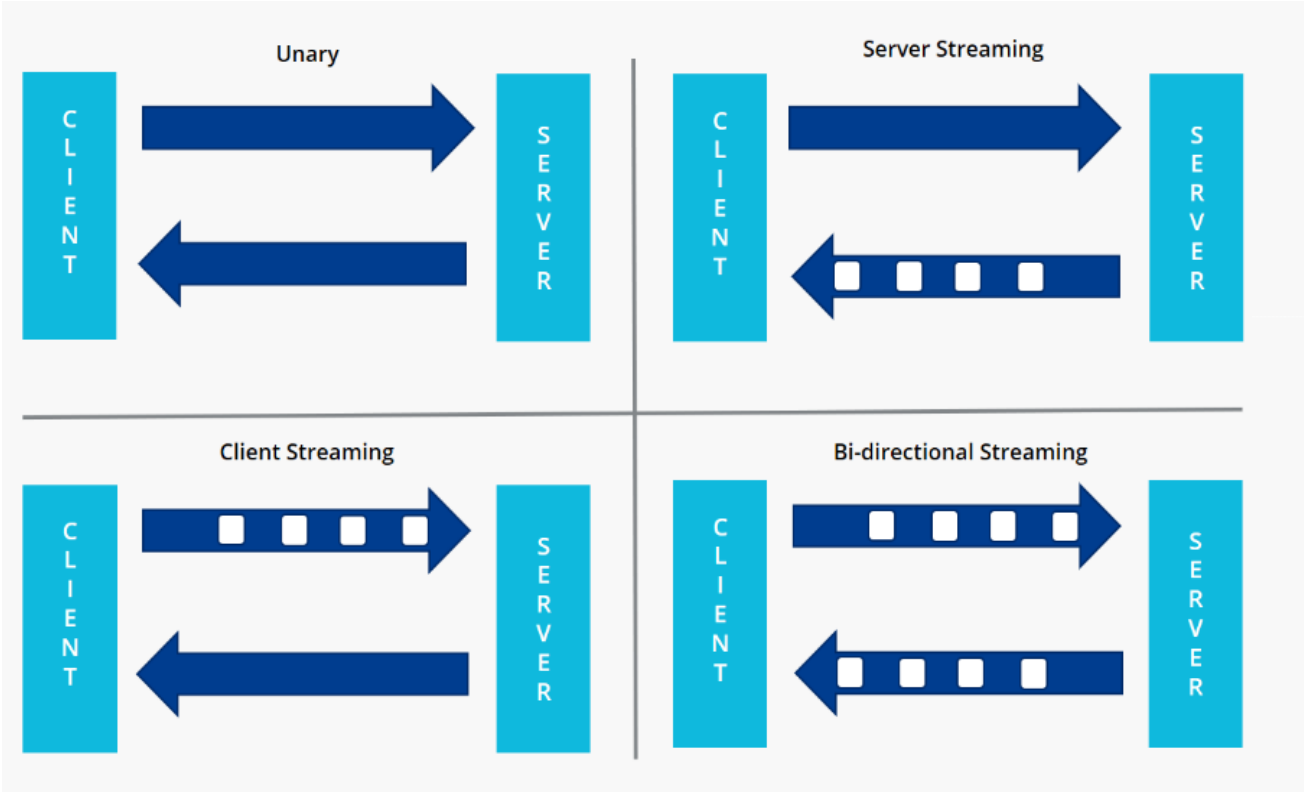
gRPC kasutuse näide



gRPC eelised

- Kiirem kui teksti põhised XML või JSON HTTP päringud
- Binaarse protokoll-puhvri sõnumid on väiksemad kui JSON/XML
  - Kui JSON võtab 81 baiti – siis grpc sõnum võtab 33
- Toetab sõnumite voogedastust

Allikas: https://www.ionos.ca/digitalguide/server/know-how/an-introduction-to-grpc/



gRPC .proto faili näide

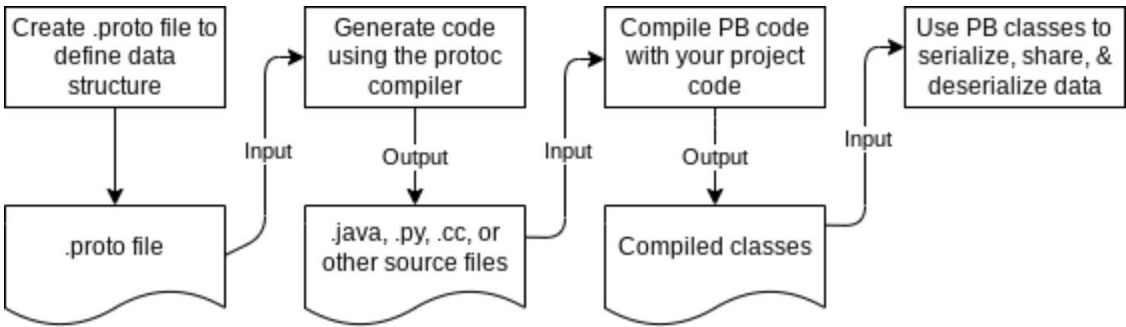
```
syntax = "proto3";
package gRPC_service;
import "google/protobuf/wrappers.proto";

service InventoryService {
  rpc getItemByName(google.protobuf.StringValue) returns (Items);
  rpc getItemByID(google.protobuf.StringValue) returns (Item);
  rpc addItem(Item) returns (google.protobuf.BoolValue);
}

message Items {
  string itemDesc = 1;
  repeated Item items = 2;
}

message Item {
  string id = 1;
  string name = 2;
```

Allikas: https://protobuf.dev/overview/



Järgmine loeng

- Jätkame andmete vahenduse ja liideste teemal
  - Hajusobjektid (RMI)
  - SOAP
  - REST
- Veebiteenused ja API'd

Selle nädala Praktikum

- Jätkame RabbitMQ kasutamist
- RPC üle RabbitMQ
  - Sarnane JSON RPC'le
  - Ehitame ise RPC!

Allikad ja viited

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