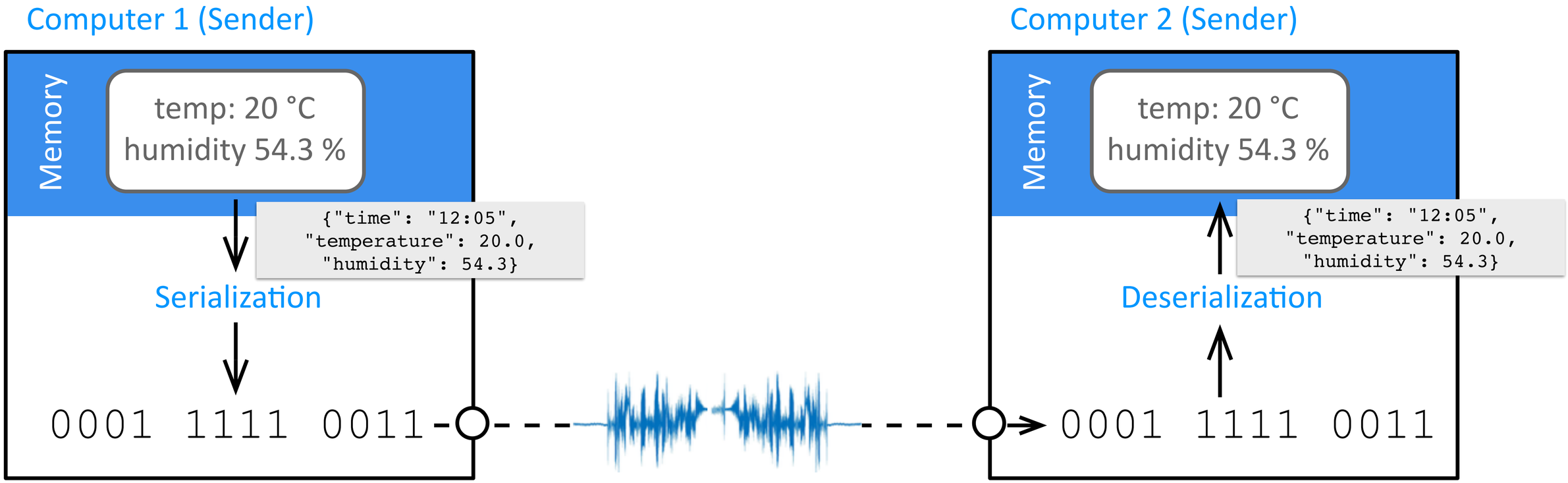
TTM4175: Introduksjon til
kommunikasjonsteknologi og digital sikkerhet

Internet of Things

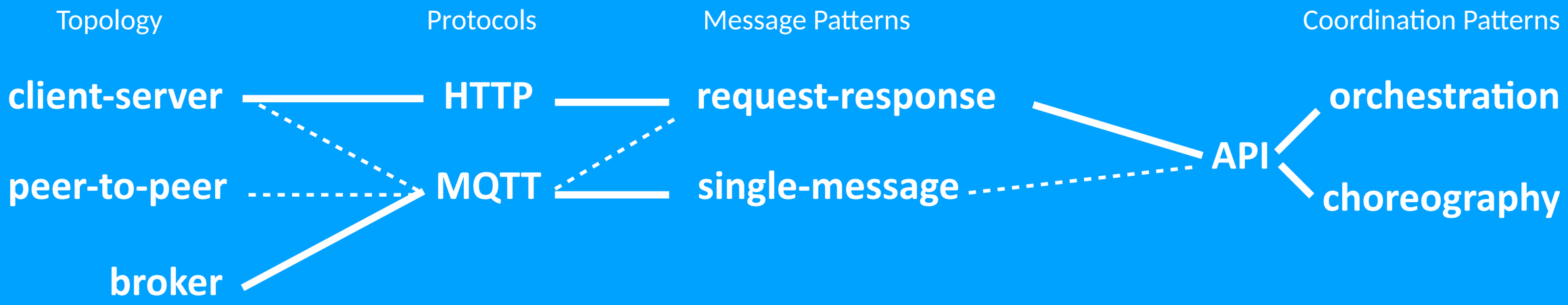
Frank Alexander Kraemer
kraemer@ntnu.no

NTNU

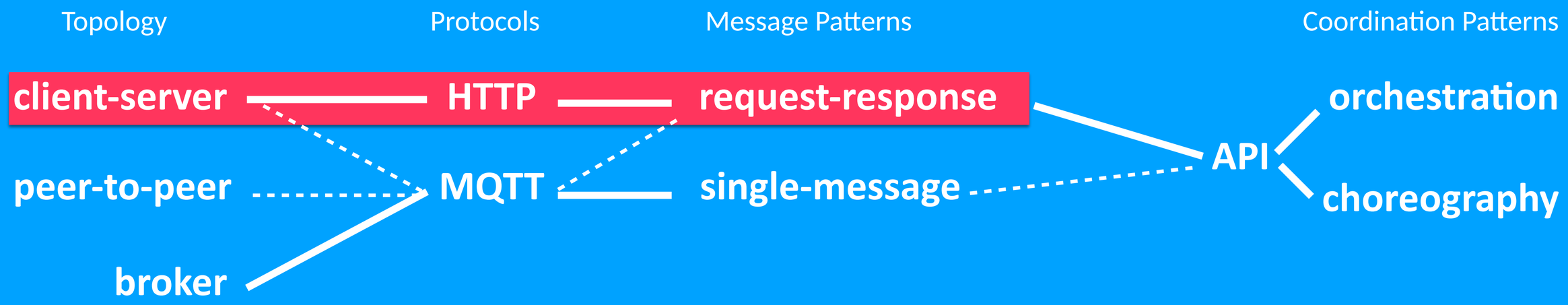
Serialization



Thinking in Patterns



Thinking in Patterns



Why are such patterns important?

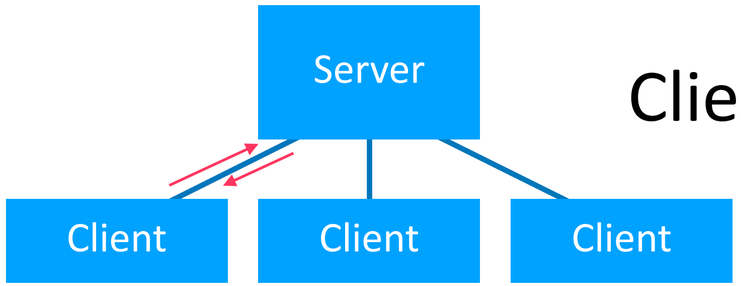
Topology, message patterns, protocols constrain certain properties of the system:

- Performance, dependability, latency, functionality,…

Wrong design choices are hard to correct

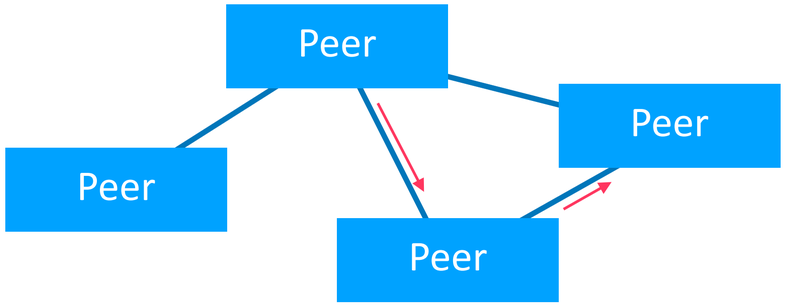
- Applications are not responsive, must be “refreshed” by the user

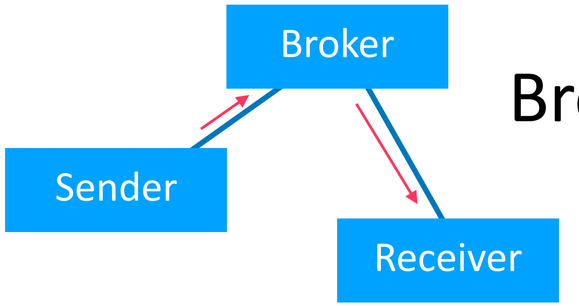
Topologies



Client-Server

Peer-to-Peer





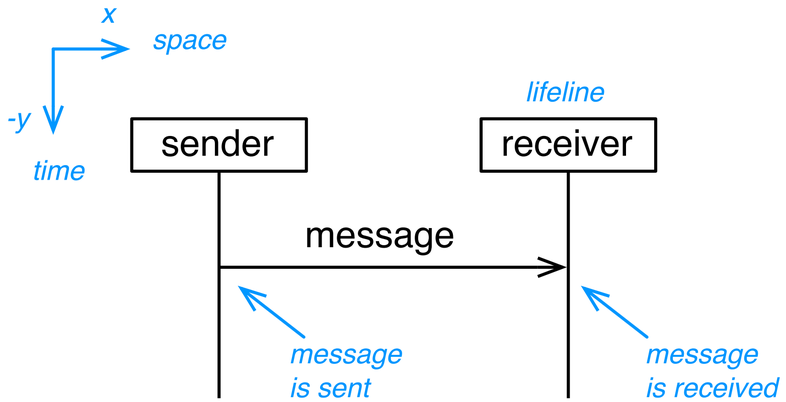
Broker

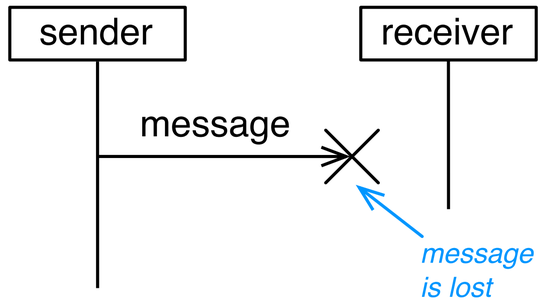
Single Message

- simplest pattern
- messages can be lost

Because the sender does not get any feedback from the receiver, we don’t know if the message arrived.

How about giving that feedback?
—> Request-response!



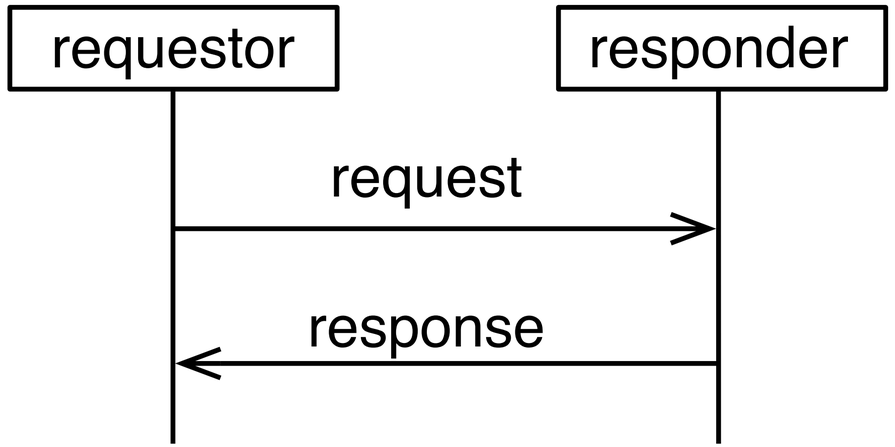


Sequence Diagrams

Show specific scenarios (examples) of message passings, between several lifelines.

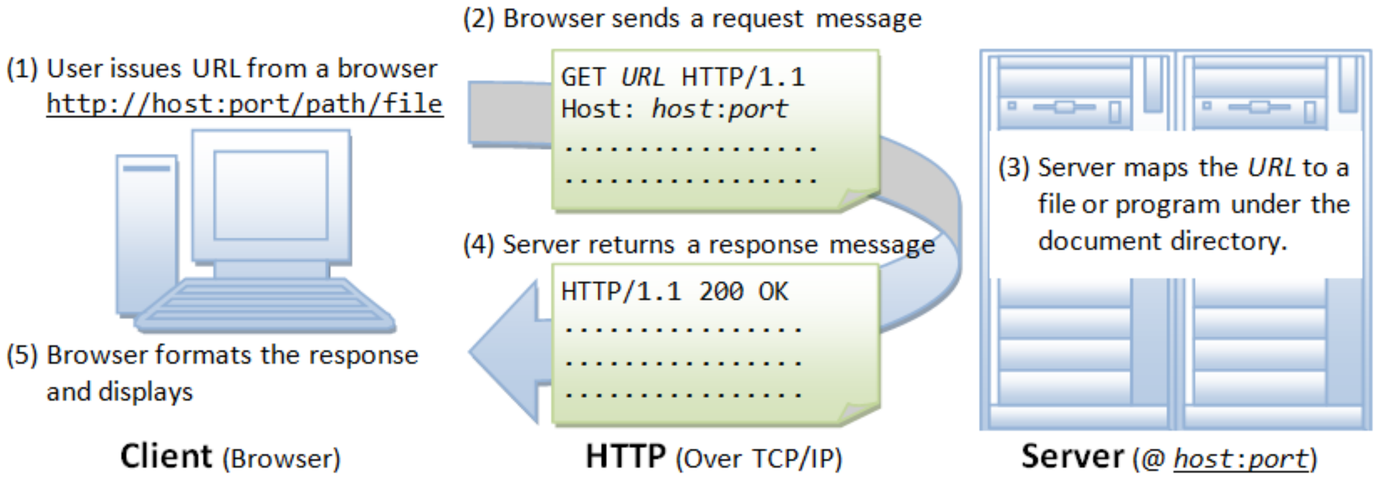
Request-Response

- Receiving the response means that the request has been received



Request-Response

- Probably the most commonly used pattern
- HTTP Requests are request-response



https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html

Reminder: Application Programming Interface (API)

- Within a **local** application,
  - for instance to access hardware, math,…
- Between components of a **distributed** application
  - that means, between our own components
- To a **remote** service
  - operated by someone else

Sanntid

Med sanntid kan du sjekke oppdatert avgangstidspunkt fra din holdeplass.

AtBs busser er utstyrt med GPS for å gi deg sanntidsinformasjon. Sanntid er en prognose for når bussen har forventet avgang fra holdeplass.

I tillegg til sanntid i reiseplanlegger, appen AtB Reise og på holdeplass kan du benytte deg av åpne data eller lage din egen sanntidsskjerm.

Lag din egen sanntidsskjerm

Åpne data

Data hentes via SIRI-protokollen: https://en.wikipedia.org/wiki/Service_Interface_for_Real_Time_Information

AtB har tatt i bruk disse elementene av SIRI-protokollen

Stop Monitoring

- http://st.atb.no/SMWS/SMService.svc?wsdl

Vehicle Monitoring

- http://st.atb.no/VMWS/VMService.svc?wsdl

Situation Exchange

- http://st.atb.no/SXWS/SXService.svc?wsdl

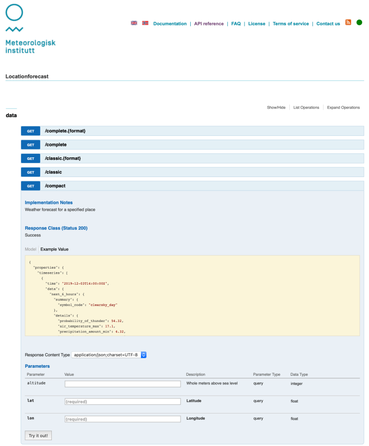
AtB har dessverre ikke anledning til å yte brukerstøtte på disse tjenestene.

Lisens vil være Norsk lisens for offentlige data (NLOD).

Weather Service Example

Meteorologisk institutt

HTTP

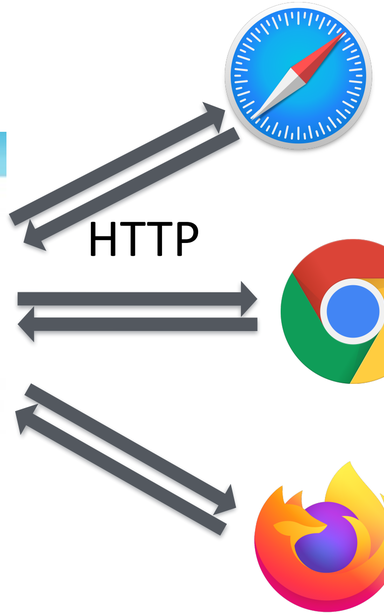


api.met.no

HTTP

yr.no

HTTP

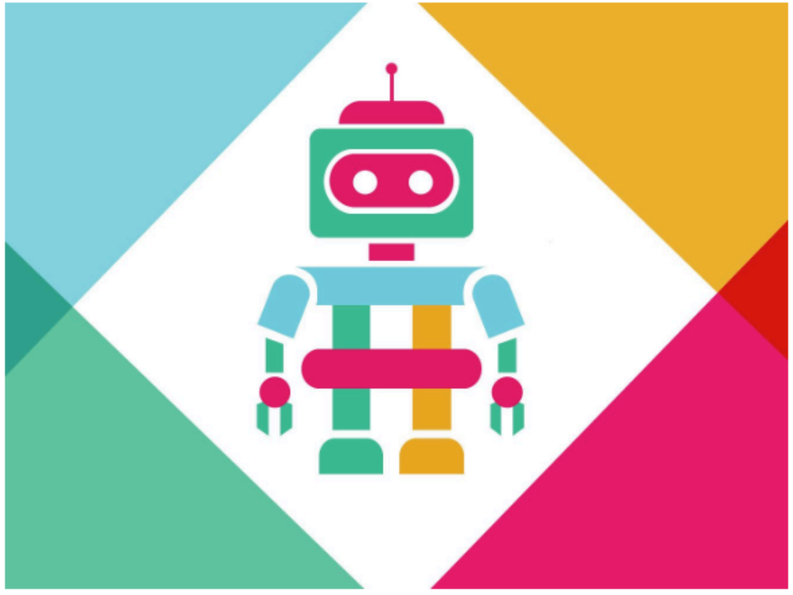


HTTP

Yr App

MOLLY MCHUGH GEAR 08.21.15 6:30 AM

SLACK IS OVERRUN WITH BOTS. FRIENDLY, WONDERFUL BOTS



THEN ONE/WIRED

THERE IS A Slack bot for everything. Oskar tracks how happy your coworkers are. Shoulda Coulda shows you how many times everyone says "should." Huskybot is "for people who need Siberian huskies, now."

Since Slack launched two years ago, it's been *the* darling app, the hot new thing for intraoffice chat and organization. It perfectly combined the long-cherished interoperability of IRC and Hipchat with the trendy polish of Trello. Microsoft

MOST POPULAR


SECURITY
Google's Chrome Hackers Are About to Upend Your Idea of Web Security
3 DAYS


INFOGRAPHICS
Let's All Obsess Over This Intricate Map of Alt Music History
10.07.16


NATIONAL AFFAIRS
Women—and Men—Share Their Harrowing Stories of Workplace Harassment
7 HOURS

MORE STORIES

Slack offer several APIs to enable different communication patterns:

Web API: Client-server. Your application is a client, taking initiative to change a Slack workspace. Your application needs to take initiative, cannot be notified directly on events.

Events API: Your application defines a web server. You tell Slack the address of the web server. Whenever something happens, Slack (as a client) calls your server.

Real Time Messaging API: Your application is a client, Slack is a server, similar to the Web API. In this API, however, the communication uses the Websockets protocol, which is bi-directional.

Pass argu

- GET
- POST parameters presented as `application/x-www-form-urlencoded`
- or a
- Some

Some me
associa
using a `lication/x-www-form-urlencoded` Content-type, and we strongly recommend
using JSO

POST

W n ser
param rs or use JSON instead.

URL-en

When ser
`applicati`
3986.

For example, a POST request to `conversations.create` might look something like this:

```
POST /api/conversations.create
Content-type: application/x-www-form-urlencoded
token=xoxa-xxxxxxxxx-xxxx&name=something-urgent
```

JSON-encoded bodies

For these write methods, you may alternatively send your HTTP POST data as `Content-type: application/json`.

There are some ground rules:

REST APIs

REST and HTTP Methods

Uniform Resource Locator (URL)	GET	PUT	PATCH	POST	DELETE
Collection https://api.example.com/resources	**List** the URIs and perhaps other details of the collection's members.	**Replace** the entire collection with another collection.	*Not generally used*	**Create** a new entry in the collection.	**Delete** the entire collection.
Element https://api.example.com/resources/item17	**Retrieve** a representation of the addressed member of the collection.	**Replace** the addressed member of the collection, or if it does not exist, **create** it.	**Update** the addressed member of the collection.	*Not generally used.*	**Delete** the addressed member of the collection.

From https://en.wikipedia.org/wiki/Representational_state_transfer

REST Principles

Client–server separation:

The client and the server act **independently**,

each on its own,

and the interaction between them is only in the form of **requests**,

initiated by the **client** only,

and responses, which the **server** sends to the client only as a reaction to a request.

The server just sits there waiting for requests from the client to come.

The server doesn’t start sending away information about the state of some resources on its own.

REST Principles

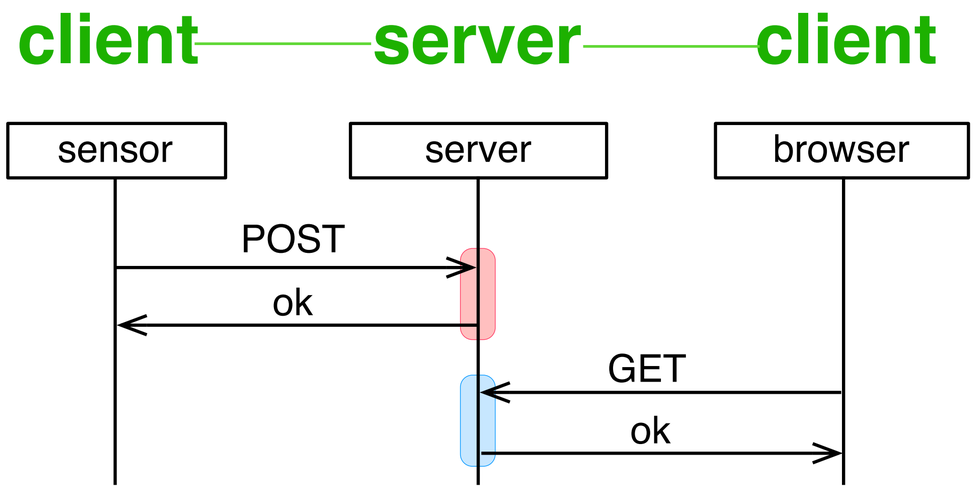
Stateless:

Stateless means the **server does not remember anything about the user** who uses the API.

It doesn’t remember if the user of the API already sent a GET request for the same resource in the past, it doesn’t remember which resources the user of the API requested before, and so on.

Each individual request contains all the information the server needs to perform the request and return a response, regardless of other requests made by the same API user.

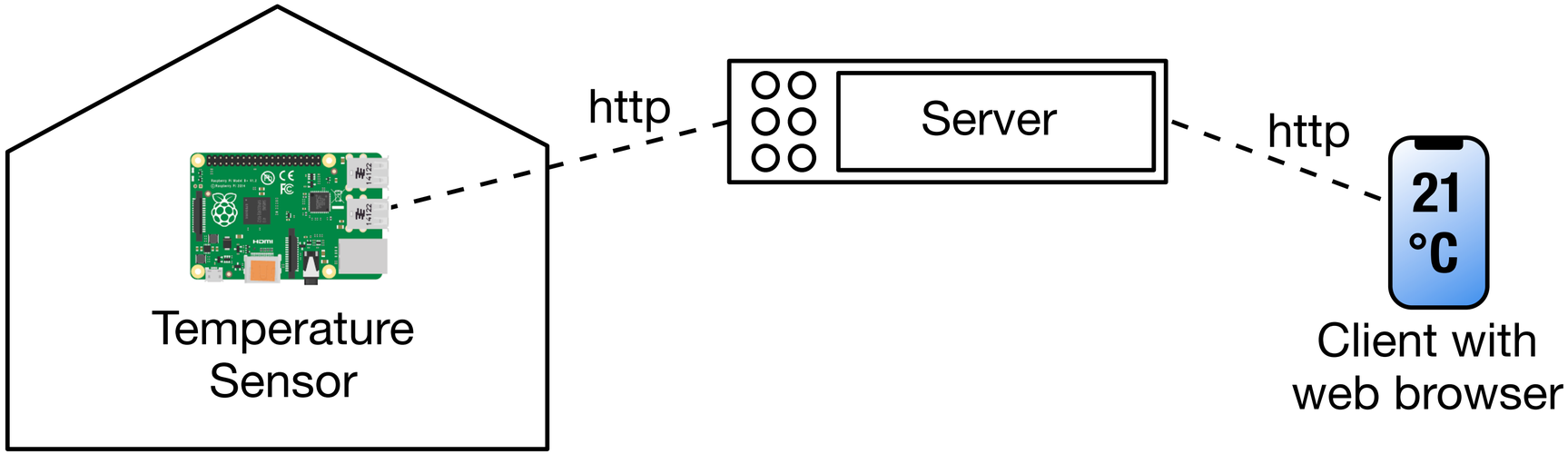
HTTP Server in Python




```
from http.server import HTTPServer, BaseHTTPRequestHandler

class RequestHandler(BaseHTTPRequestHandler):
        def do_POST(self):
            # parse request
            # work on data (store, get)
            # create response

        def do_GET(self):
                # parse request
                # work on data (store, get)
                # create response

server_address = ('', 8000)
httpd = HTTPServer(server_address, RequestHandler)
httpd.serve_forever()
```

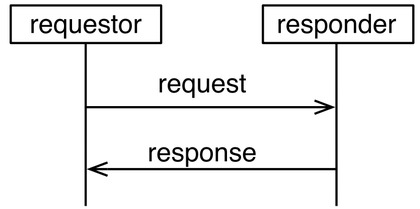
NTNU

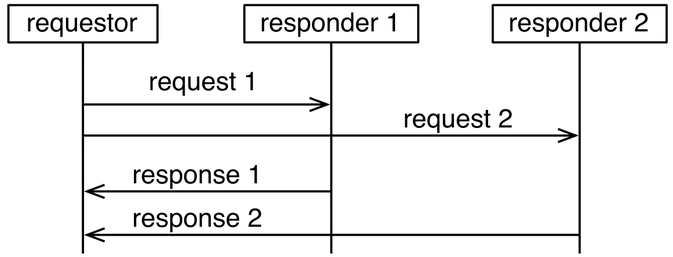
Overall Architecture

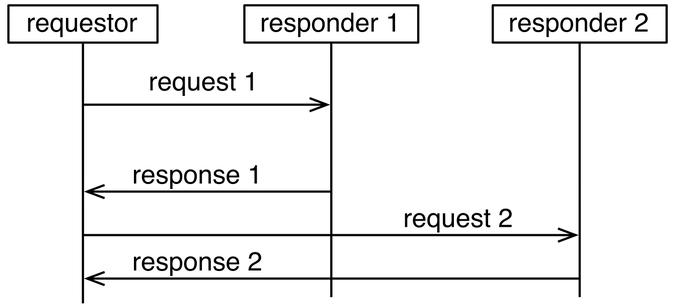


Request-Response

- Synchronous and asynchronous:
  - when several requests can happen at the same time —> asynchronous



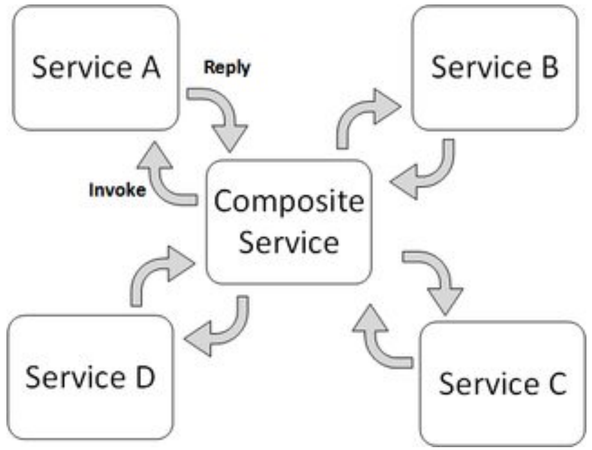


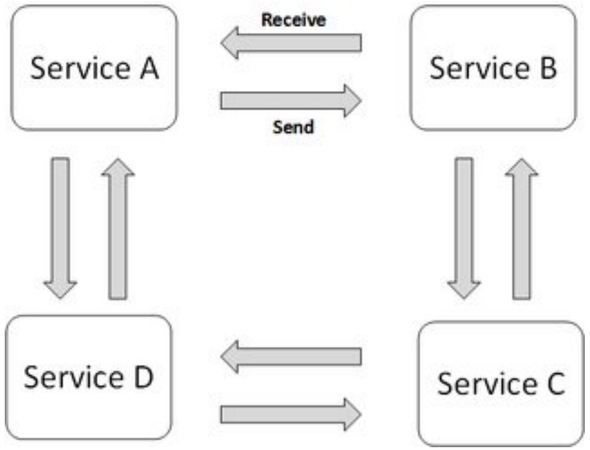


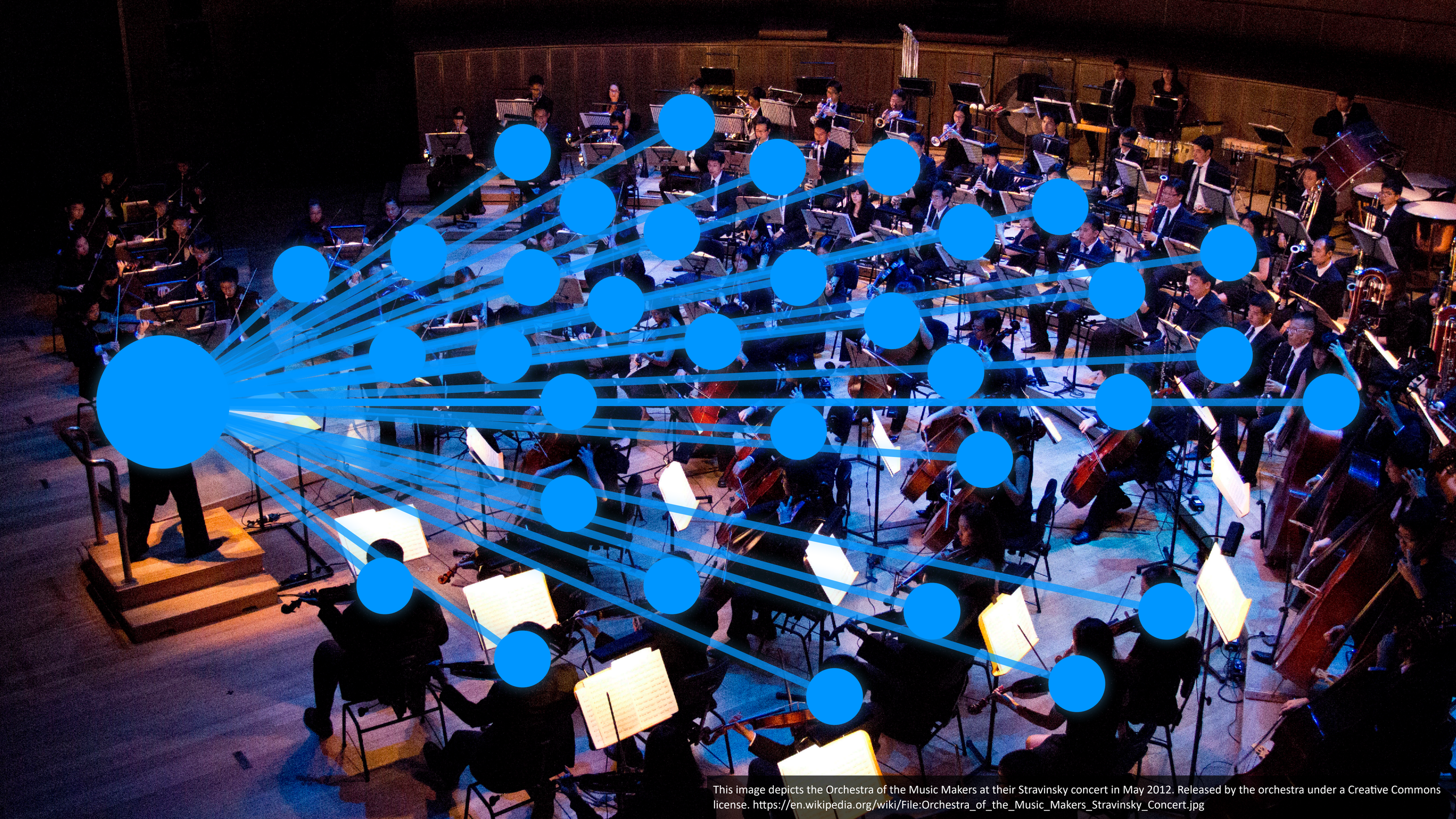
Orchestration and Choreographies

Orchestration and Choreography

- **Problem:** We need to coordinate a set of services (A, B, C, D) so that they together produce a result.

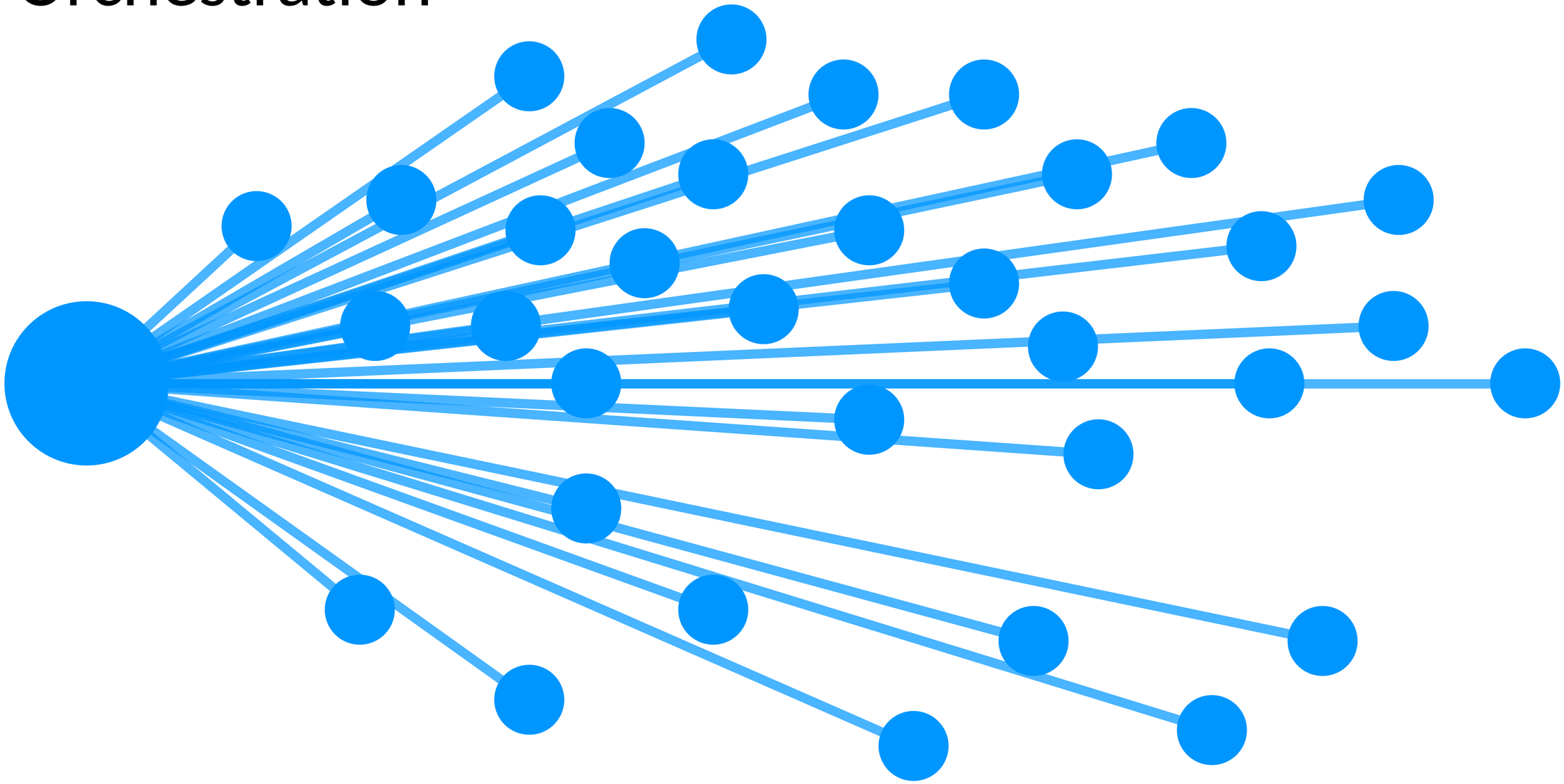






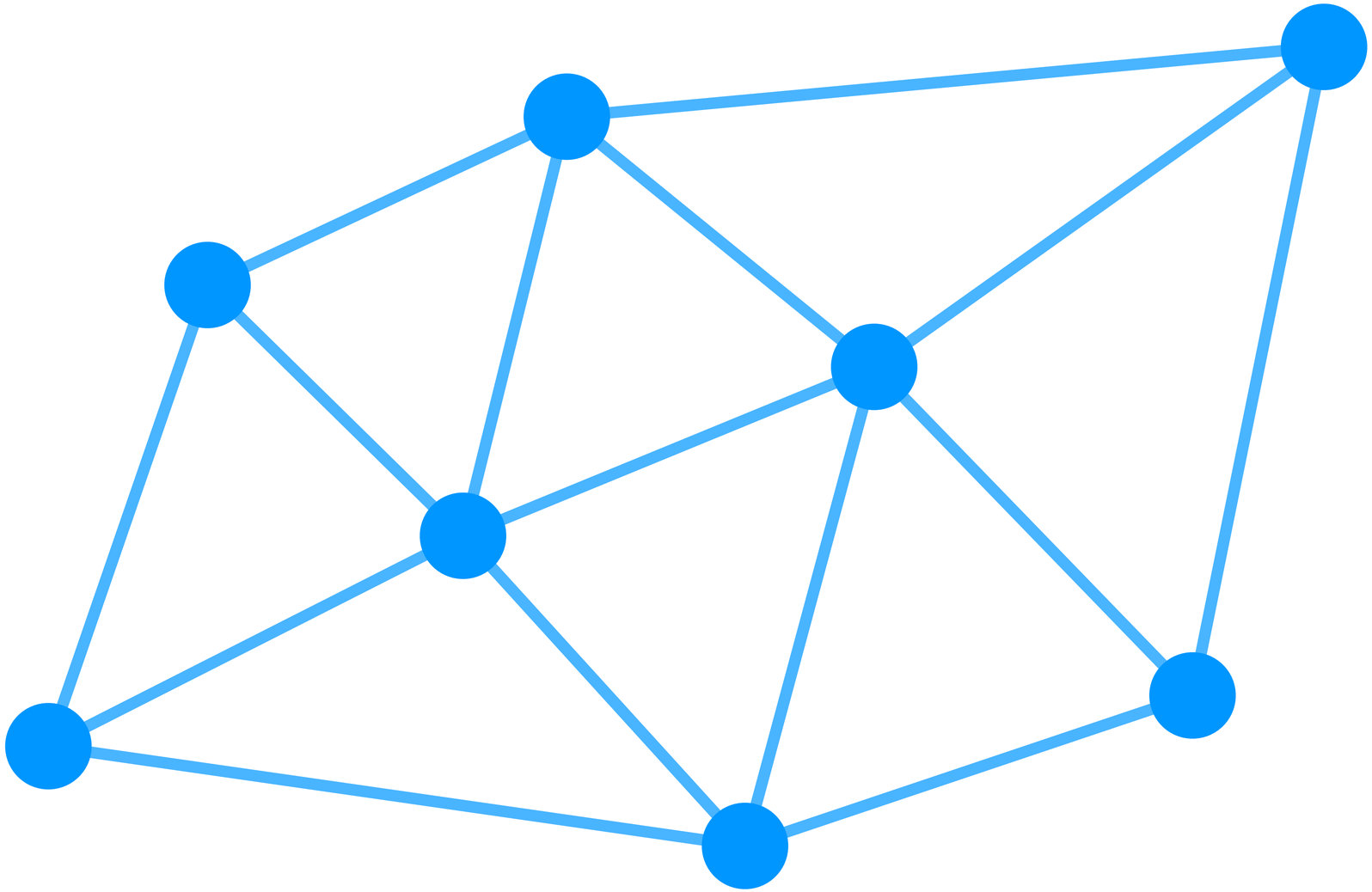
This image depicts the Orchestra of the Music Makers at their Stravinsky concert in May 2012. Released by the orchestra under a Creative Commons license. https://en.wikipedia.org/wiki/File:Orchestra_of_the_Music_Makers_Stravinsky_Concert.jpg

Orchestration

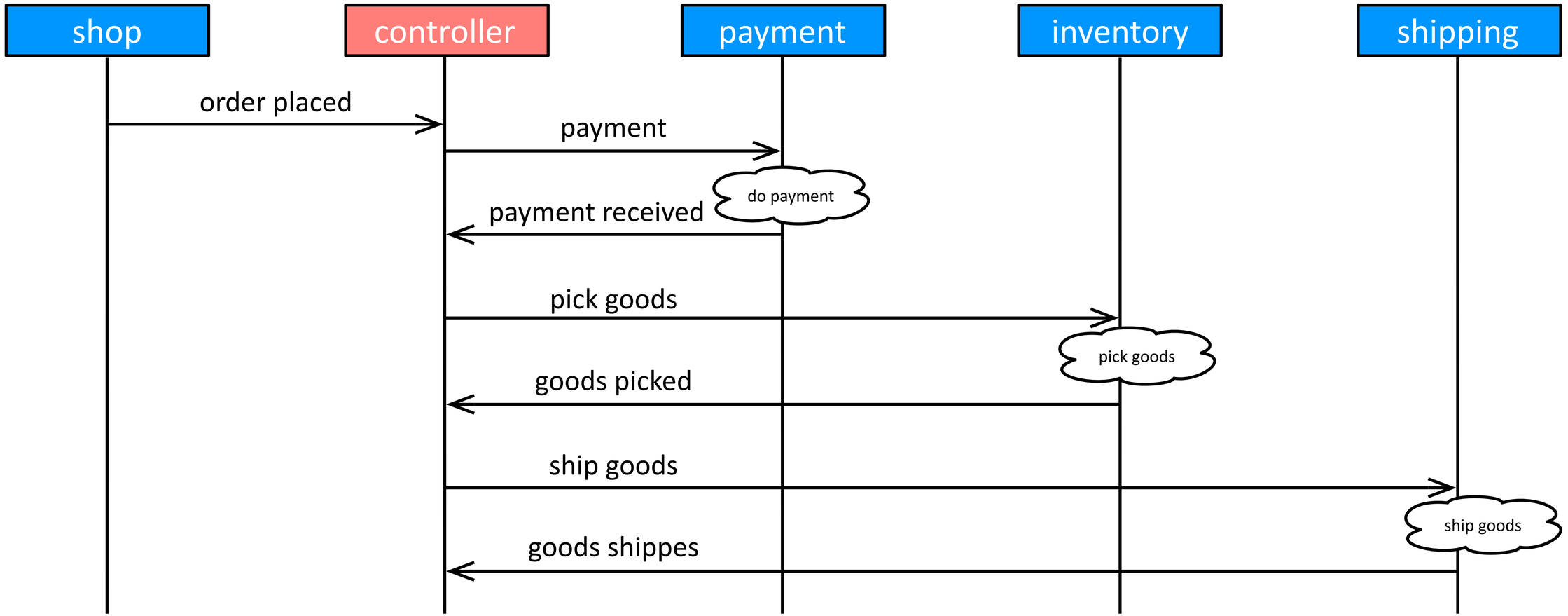


https://pixabay.com/en/girls-water-pool-swim-girl-float-2526556/

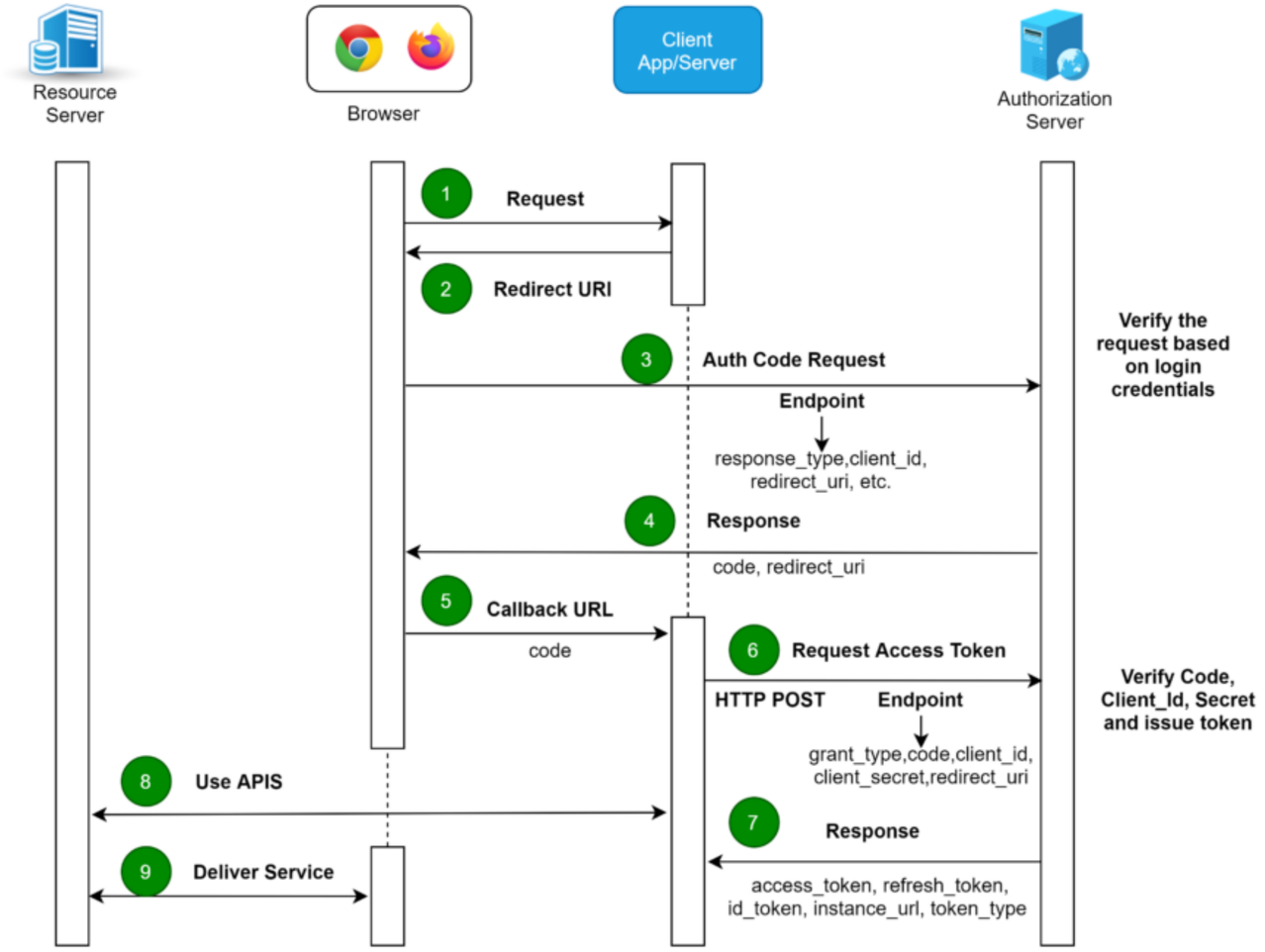
Choreography



https://commons.wikimedia.org/wiki/File:Miami_City_Ballet_%5E_The_Cleveland_Orchestra_-_Flickr_-_Knight_Foundation_(2).jpg


shop
controller
payment
inventory
shipping
order placed
payment
do payment
payment received
pick goods
pick goods
goods picked
ship goods
ship goods
goods shippes

OAuth Example



OAuth 2.0 Web-Server Flow

NTNU