# Microservice Architecture

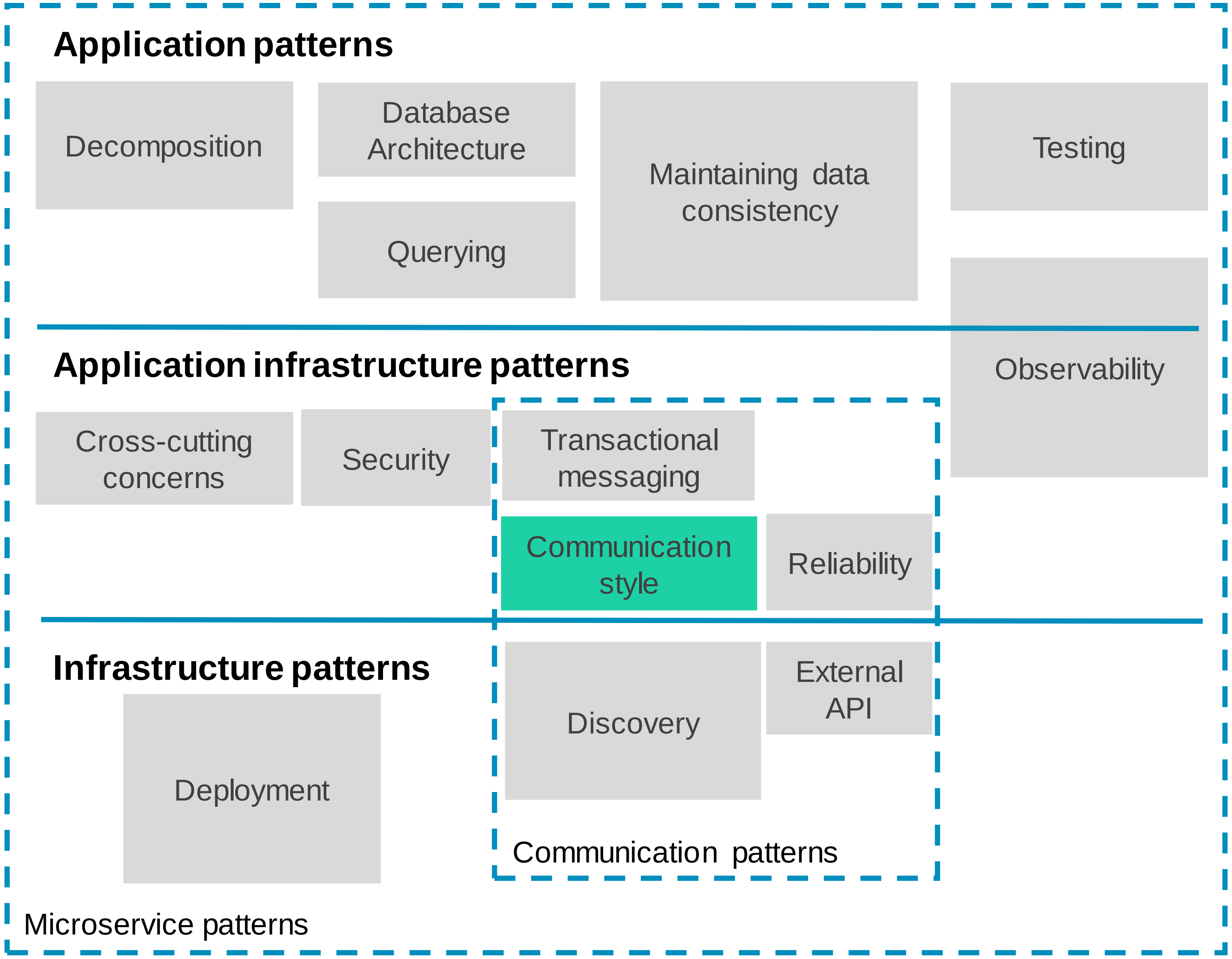
## IPC - Synchronous communication (part 1/3)

## Remote Procedure Invocation (RPI)

## By the end of this course, you will be able to

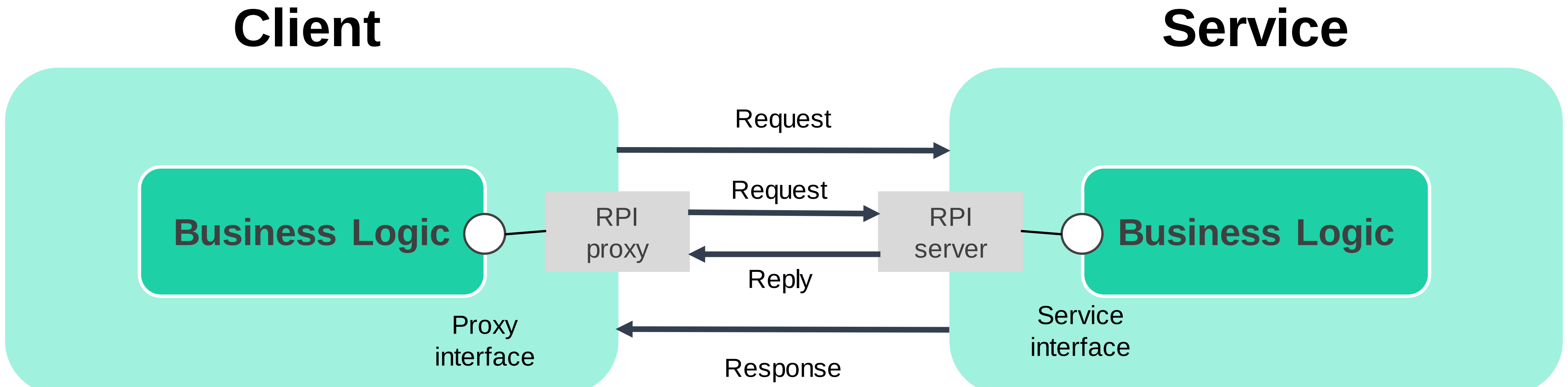
**Use** a specific technologies to **implement**
the **Synchronous Remote Procedure Invocation pattern**.

# Problem areas to solve



# Synchronous Communication

## The **Remote Procedure Invocation (RPI)** pattern



Request / Response

# Agenda

**Communicating using the Synchronous Remote Procedure Invocation pattern**

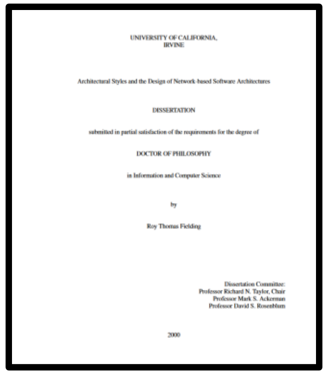
1. Synchronous communications using **REST**

2. Synchronous Communication using **gRPC**

# Synchronous Communication

## using REST

**REST** *(REpresentational State Transfer)*



UNIVERSITY OF CALIFORNIA,
IRVINE

Architectural Styles and the Design of Network-based Software Architectures

DISSERTATION

submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in Information and Computer Science

by

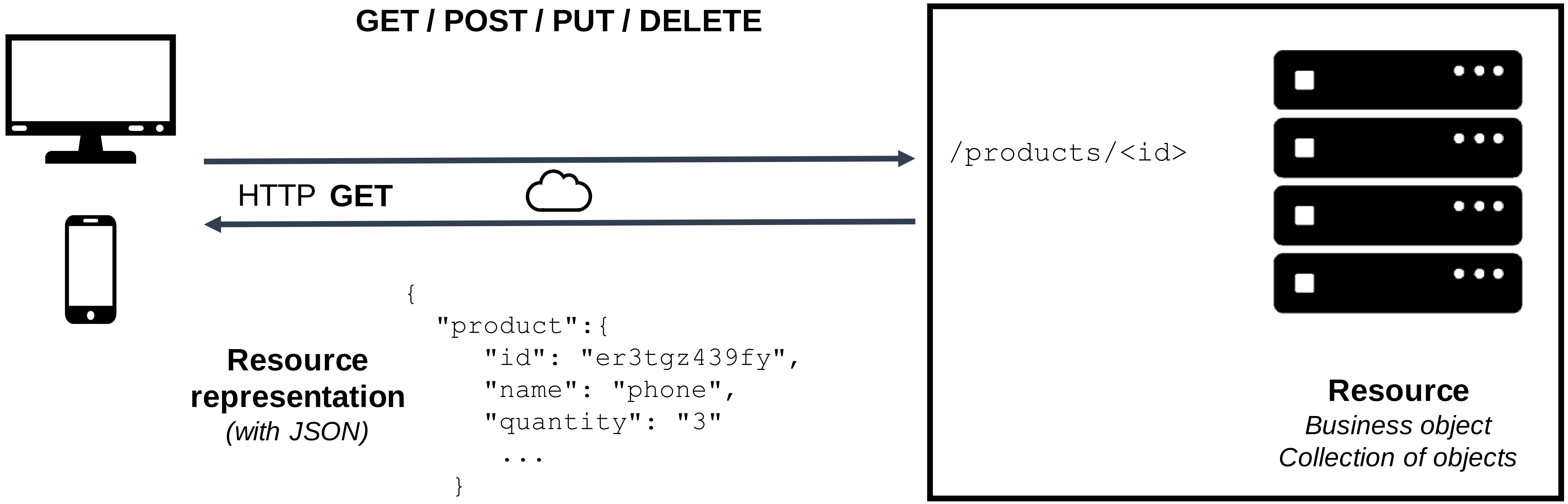
Roy Thomas Fielding

Dissertation Committee:
Professor Richard N. Taylor, Chair
Professor Mark S. Ackerman
Professor David S. Rosenblum

2000

PhD. dissertation

HTTP verbs
**GET / POST / PUT / DELETE**



**Resource representation**
*(with JSON)*

```
{
  "product":{
     "id": "er3tgz439fy",
     "name": "phone",
     "quantity": "3"
       ...
    }
}
```

# Synchronous Communication

## using REST

### **REST** *constraints*

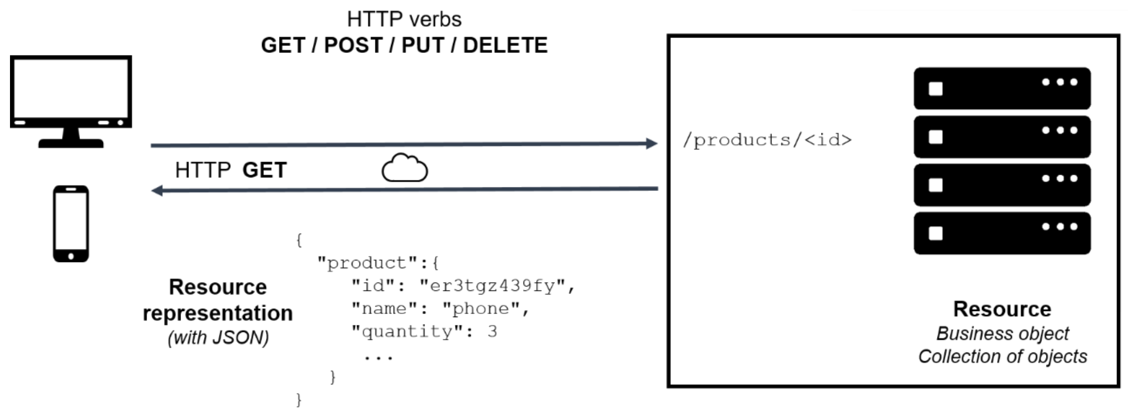
**1** Client–server architecture

**2** Statelessness

**3** Cacheability

**4** Layered system

**5** Uniform interface



```
{
  "product":{
     "id": "er3tgz439fy",
     "name": "phone",
     "quantity": 3
     ...
  }
}
```

✓ Performance ✓ Scalability ✓ Visibility

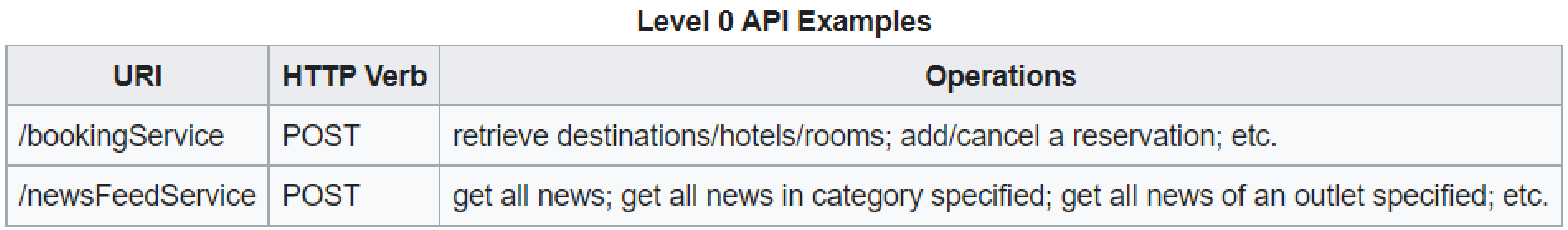
✓ Portability ✓ Reliability ...

- **Resource identification in requests**
- **Resource manipulation through representations**
- **Self-descriptive messages**
- **Hypermedia as the engine of application state (*HATEOAS*)**

# Synchronous Communication

## using REST

## The **REST** maturity model



Level 0 API Examples

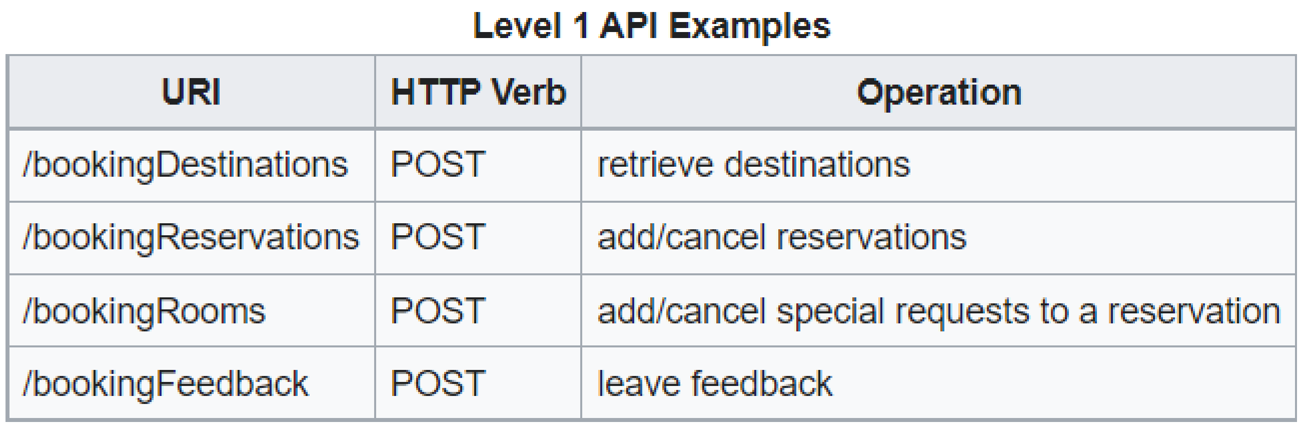
| URI | HTTP Verb | Operations |
|---|---|---|
| /bookingService | POST | retrieve destinations/hotels/rooms; add/cancel a reservation; etc. |
| /newsFeedService | POST | get all news; get all news in category specified; get all news of an outlet specified; etc. |

**Level 0:** The Swamp of POX

# Synchronous Communication

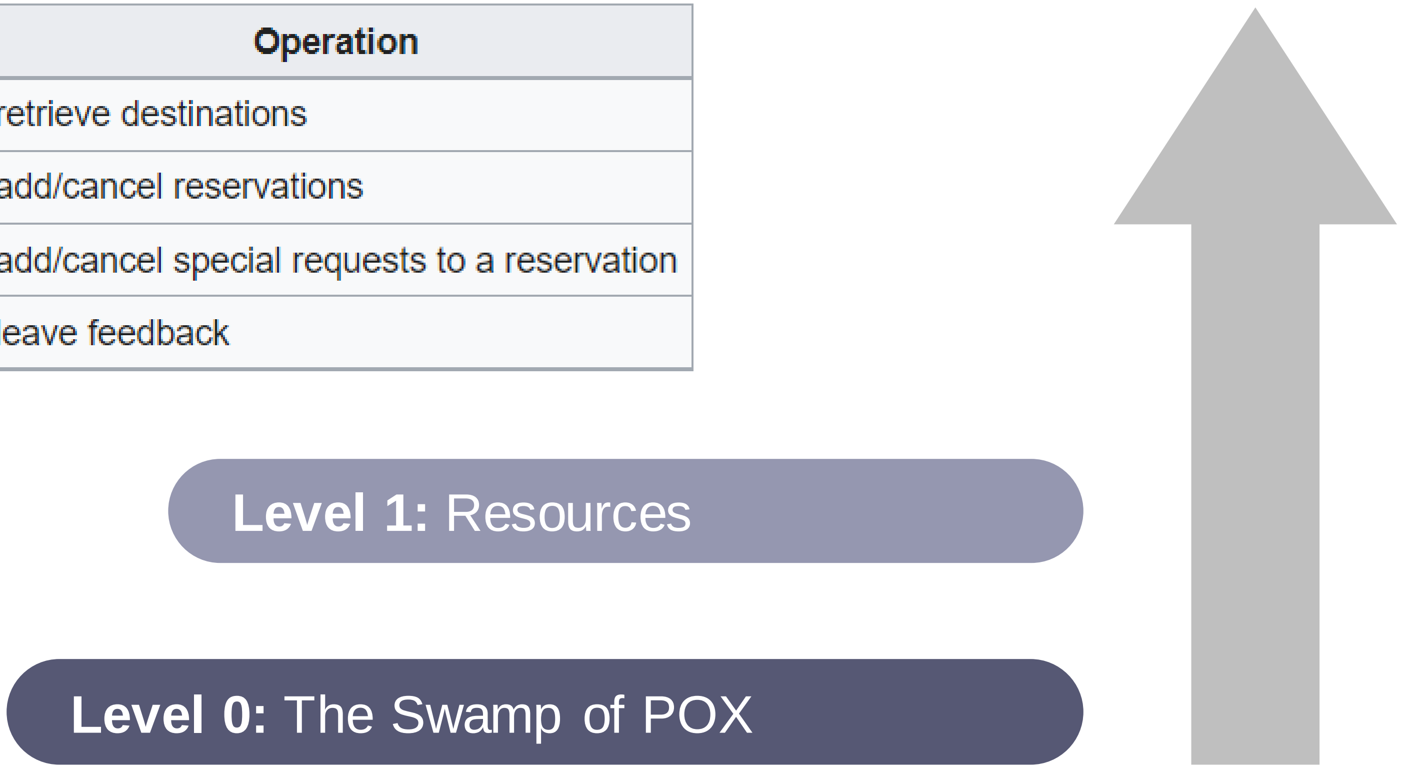
## using REST

### The **REST** maturity model



**Level 1 API Examples**

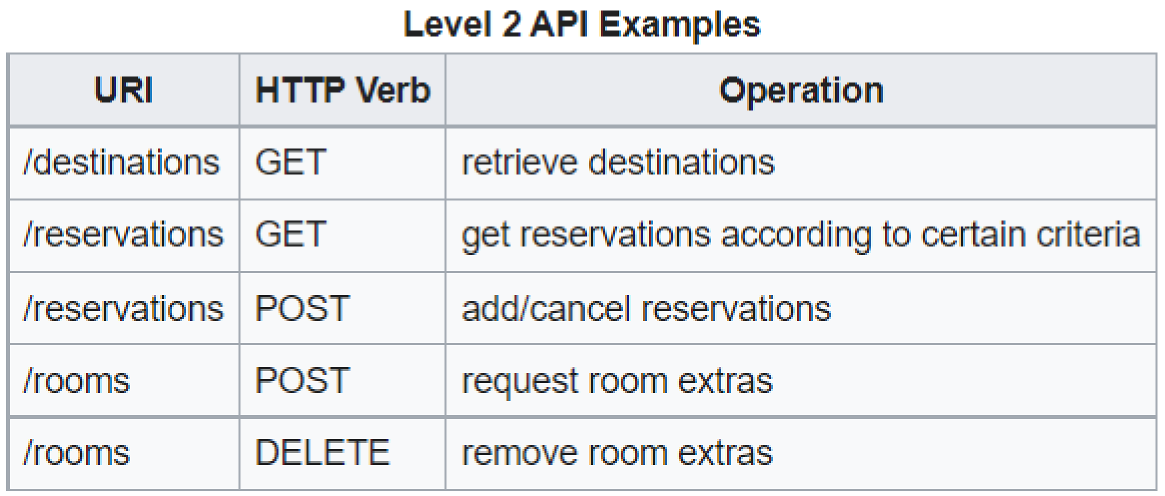
| URI | HTTP Verb | Operation |
|---|---|---|
| /bookingDestinations | POST | retrieve destinations |
| /bookingReservations | POST | add/cancel reservations |
| /bookingRooms | POST | add/cancel special requests to a reservation |
| /bookingFeedback | POST | leave feedback |



# Synchronous Communication

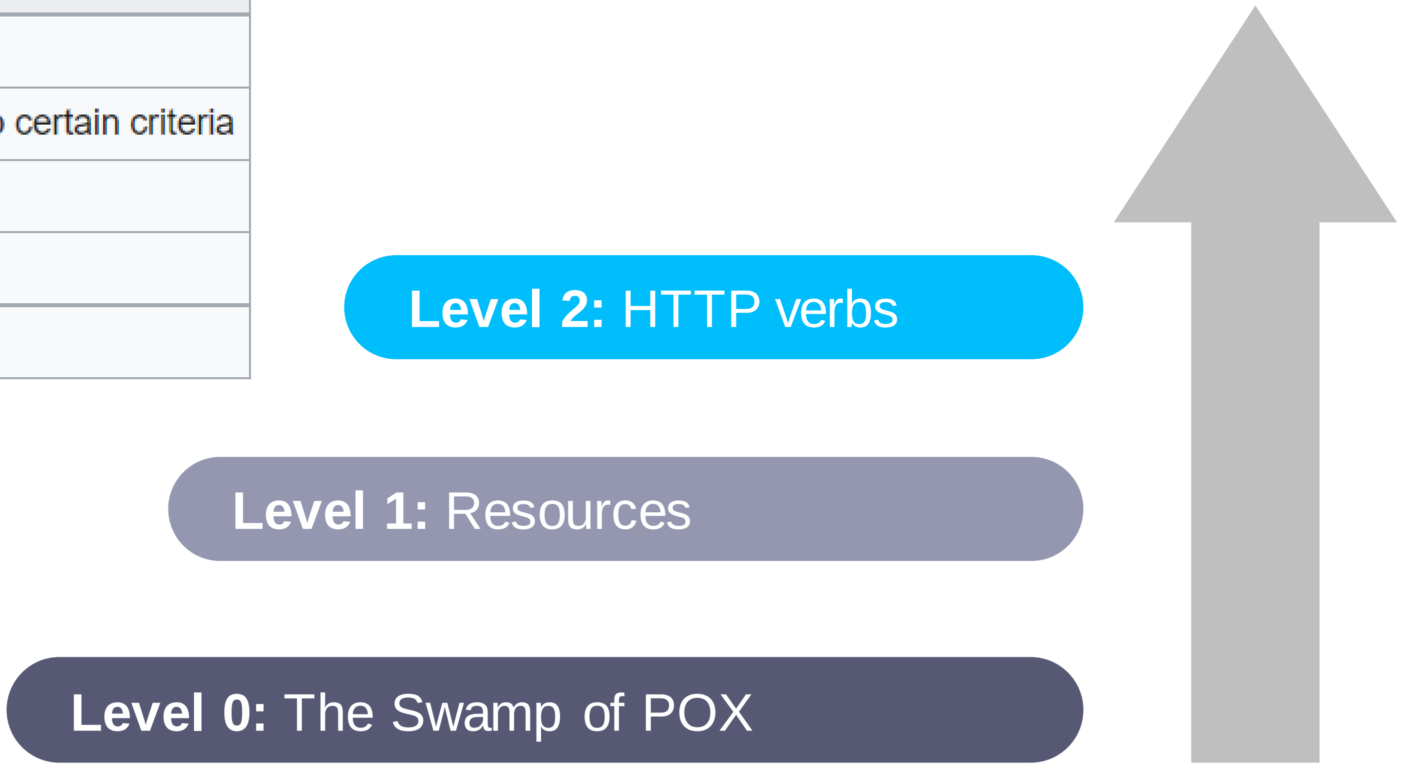
## using REST

### The **REST** maturity model



**Level 2 API Examples**

| URI | HTTP Verb | Operation |
|---|---|---|
| /destinations | GET | retrieve destinations |
| /reservations | GET | get reservations according to certain criteria |
| /reservations | POST | add/cancel reservations |
| /rooms | POST | request room extras |
| /rooms | DELETE | remove room extras |



# Synchronous Communication

## using REST

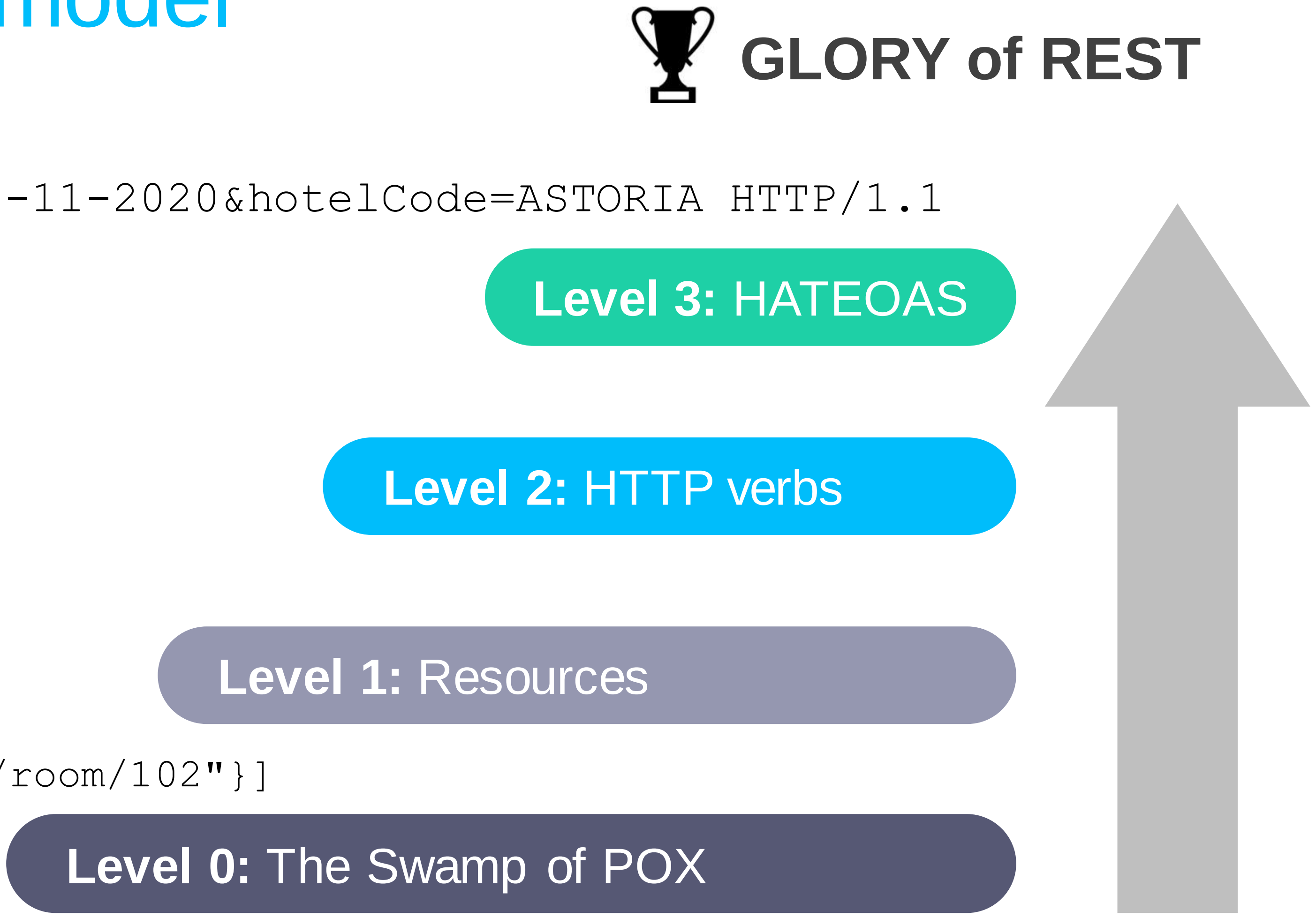
## The **REST** maturity model

**Request**

```
GET /room/?customerId=1&date=10-11-2020&hotelCode=ASTORIA HTTP/1.1
```

**Response**

```
{
    customerId: "1",
    reservations: [{
         room: "102",
         checkin: "10-11-2020",
         checkout: "11-14-2020",
         price: "100",
         href: "https://localhost:8080/room/102"}]
}
```



# Synchronous Communication

## using REST

## Specifying **REST** APIs

Open API Specification

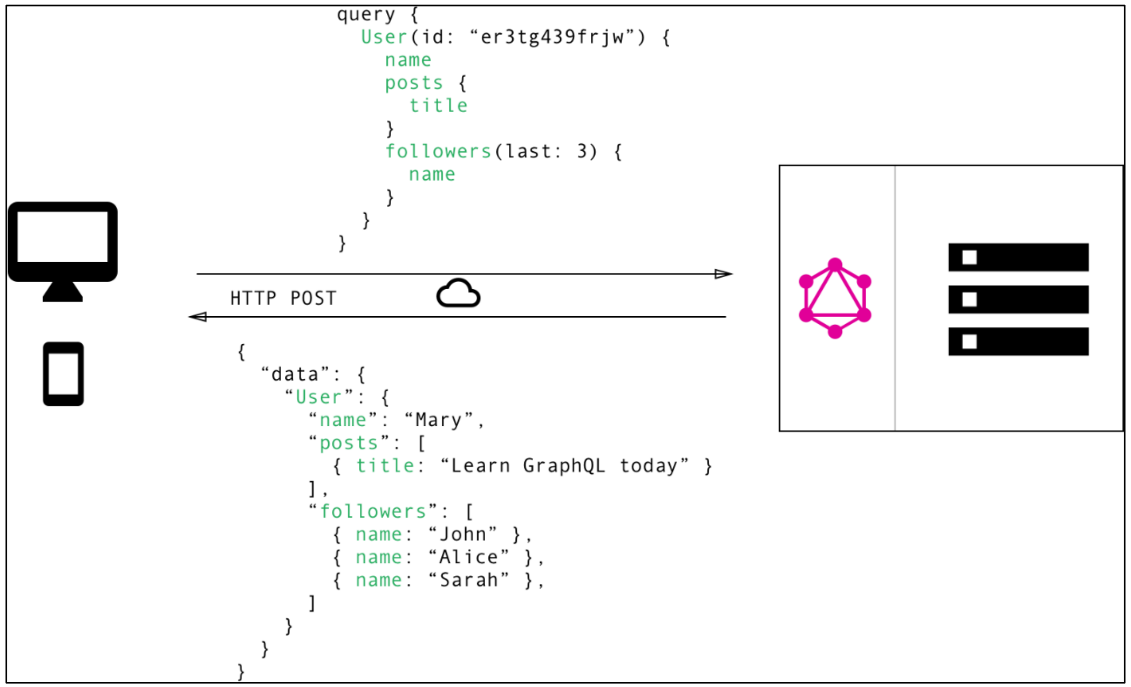
Swagger

# Synchronous Communication

## using REST

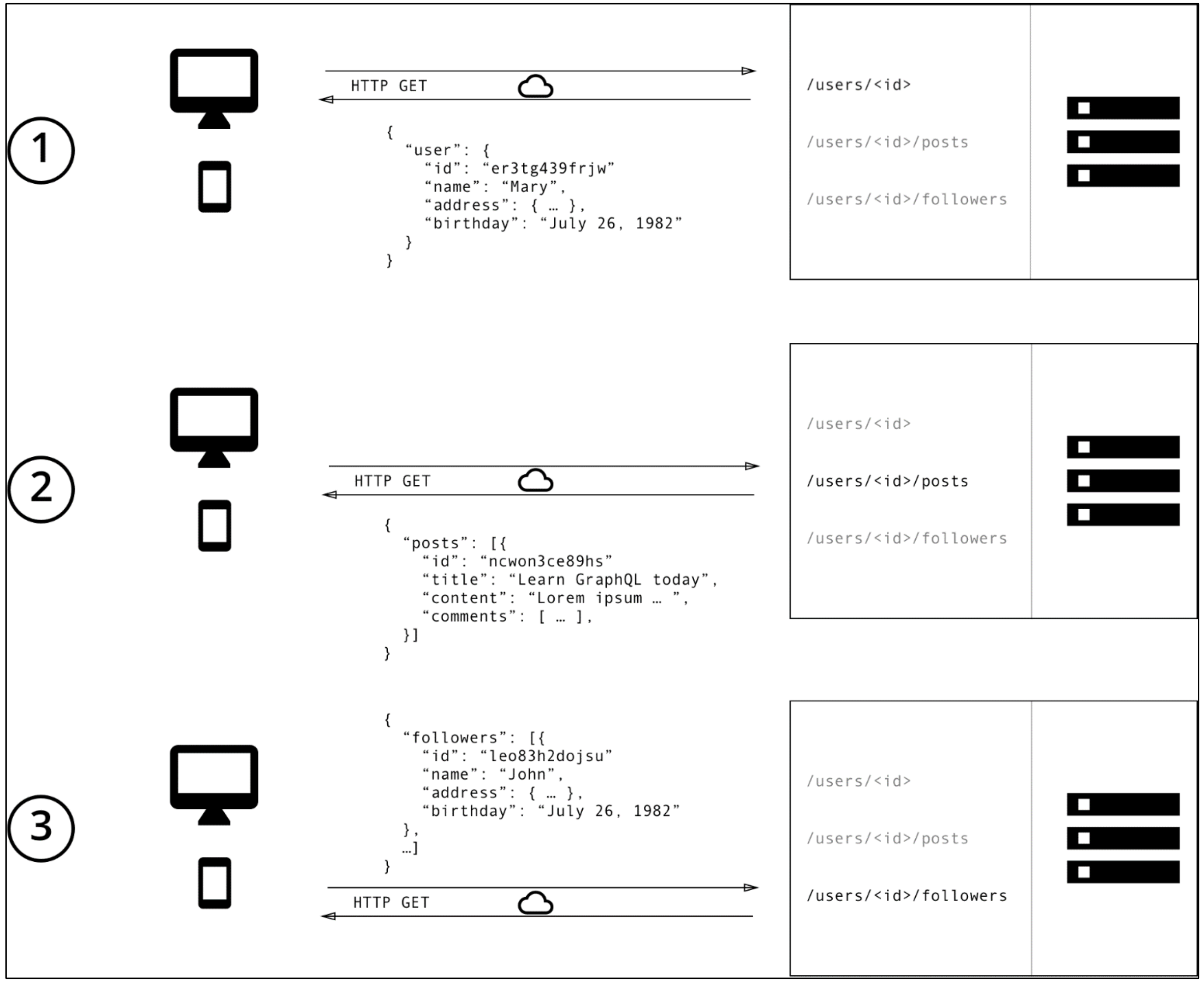
## Some challenges

**Challenge 1:** Fetching multiple resources in a single request



```
query {
  User(id: "er3tg439frjw") {
    name
    posts {
      title
    }
    followers(last: 3) {
      name
    }
  }
}
```

```
{
  "data": {
    "User": {
      "name": "Mary",
      "posts": [
        { title: "Learn GraphQL today" }
      ],
      "followers": [
        { name: "John" },
        { name: "Alice" },
        { name: "Sarah" },
      ]
    }
  }
}
```



```
{
  "user": {
    "id": "er3tg439frjw"
    "name": "Mary",
    "address": { … },
    "birthday": "July 26, 1982"
  }
}
```

```
{
  "posts": [{
    "id": "ncwon3ce89hs"
    "title": "Learn GraphQL today",
    "content": "Lorem ipsum … ",
    "comments": [ … ],
  }]
}
```

```
{
  "followers": [{
    "id": "leo83h2dojsu"
    "name": "John",
    "address": { … },
    "birthday": "July 26, 1982"
  },
  …]
}
```

# Synchronous Communication

using REST

## Some challenges

### Challenge 2: Mapping operations to http verbs

| HTTP Methods | Safe | Idempotent |
|---|---|---|
| GET | ✓ | ✓ |
| POST | ✗ | ✗ |
| PUT | ✗ | ✓ |
| DELETE | ✗ | ✓ |
| OPTIONS | ✓ | ✓ |
| HEAD | ✓ | ✓ |

**Operations** $\longleftrightarrow$ **HTTP verbs**

**Update** **?** **PUT**

**Safe**
An HTTP method is considered safe if the request made with it does not cause any side effects and does not change the state of the resource.

**Idempotent**
If you get the same response no matter how many times you request it, the method is said to be idempotent.

# Synchronous Communication

## using REST

### Benefits & Drawbacks

- ⊕ Simple and familiar
- ⊕ Testable: Easy to test an HTTP API
- ⊕ Direct support of the “request/response” communication style
- ⊕ HTTP is firewall friendly
- ⊕ No intermediate broker

- ⊖ Only support the “request/response” interaction style
- ⊖ Reduced availability
- ⊖ Clients must know the locations (URLs) of the service instances(s)
- ⊖ Fetching multiple resources in a single request
- ⊖ difficult to map multiple update operations to HTTP verbs

2

## Agenda

**Communicating using the Synchronous Remote Procedure Invocation pattern**

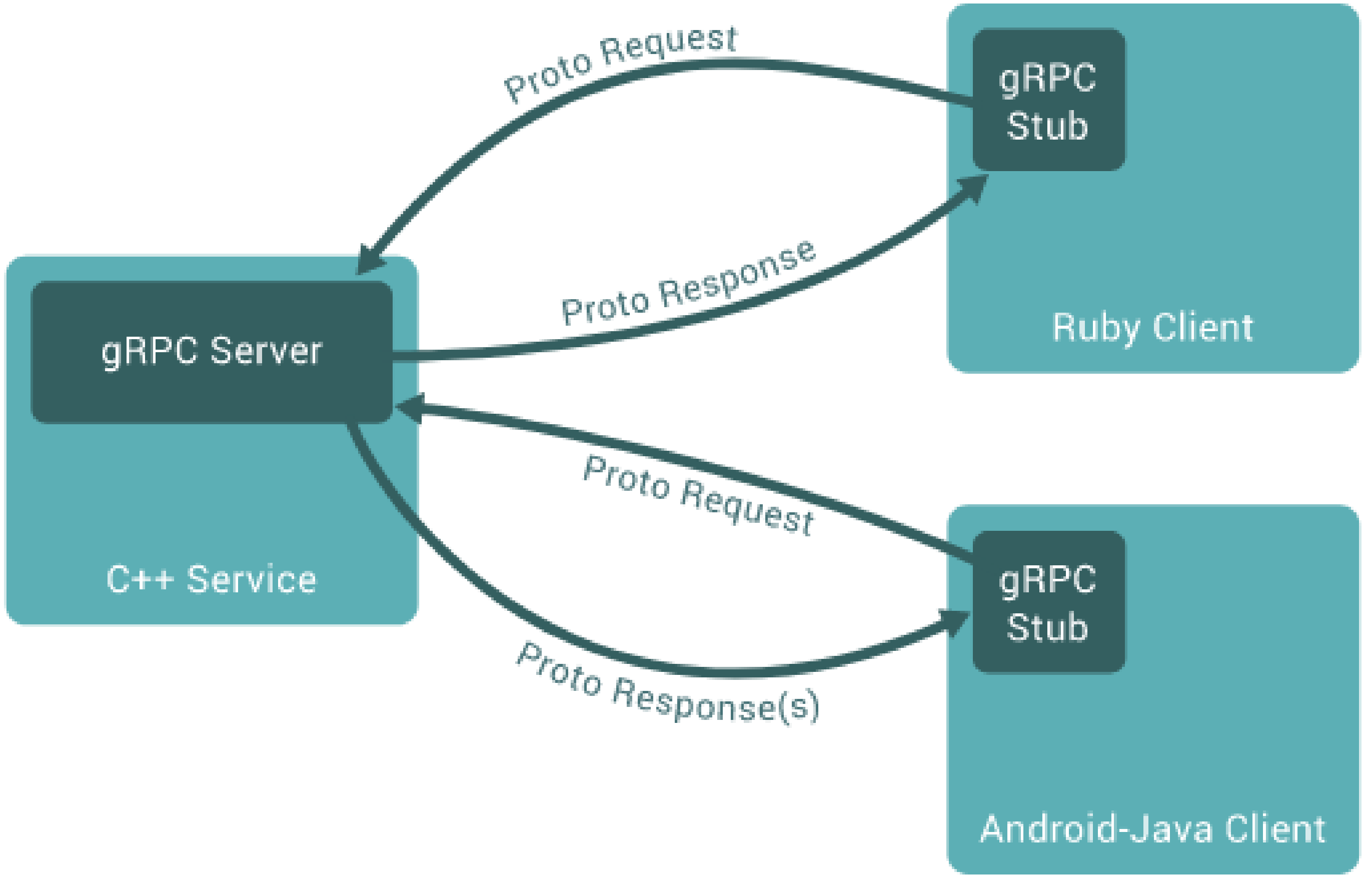
1. Synchronous communications using **REST**
2. Synchronous Communication using **gRPC**

# Synchronous Communication

## using gRPC

### **gRPC:** the principle

gRPC

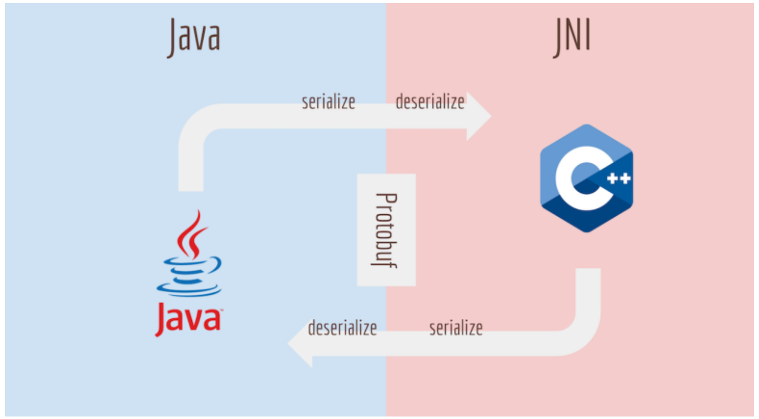


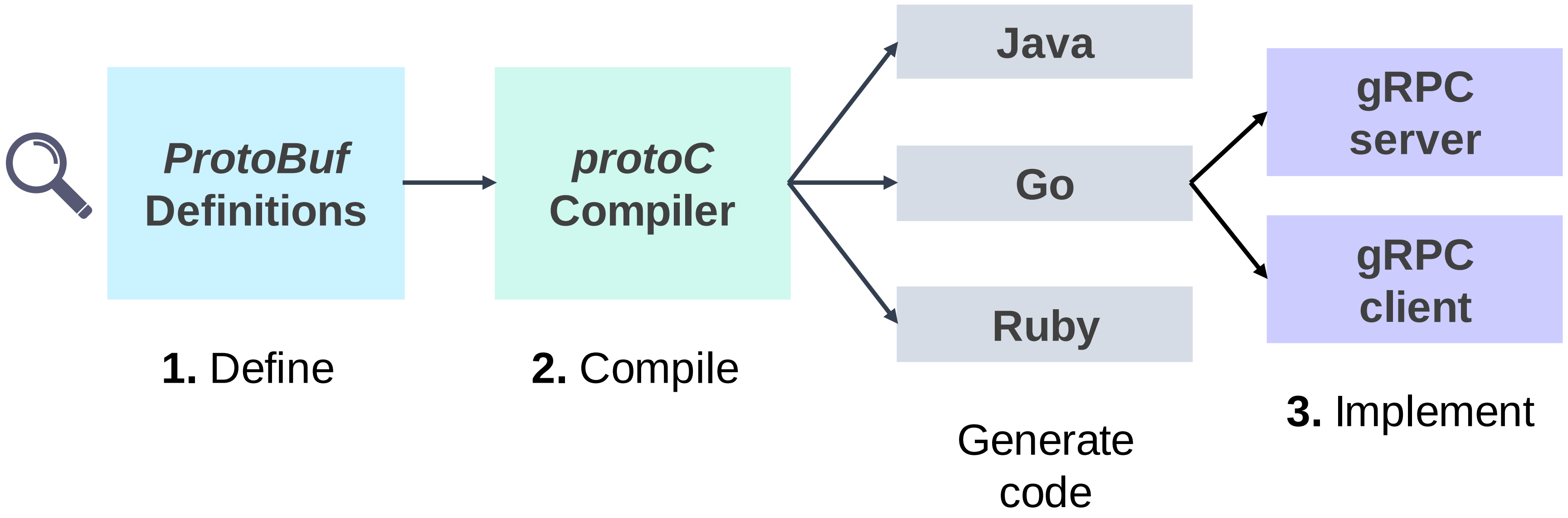
# Synchronous Communication

## using gRPC

### gRPC workflow

Protocol Buffers





# Synchronous Communication

## using gRPC

### Benefits & Drawbacks

- ⊕ Efficient and compact IPC mechanism
- ⊕ Rich set of update operations
- ⊕ Bidirectional streaming enables both RPI and messaging styles of communication
- ⊕ enables interoperability between clients and services written in a wide range of languages

- ⊖ More work to consume gRPC-based API (in some clients)
- ⊖ Old firewalls don’t support HTTP/2

## Remote Procedure Invocation pattern

- One evident and easy IPC option is using the **Synchronous Remote Procedure Invocation** (*RPI*) pattern.
- The RPI pattern can be implemented using different communication technologies including **REST** as a *de facto*.
- The REST architecture style has a **lot of benefits** and a lot of **challenges** to overcome.
- Many other projects appears to overcome such problems. The most famous are **GraphQL** and **gRPC** *(each of them has different trade-offs)*.

Questions are welcome