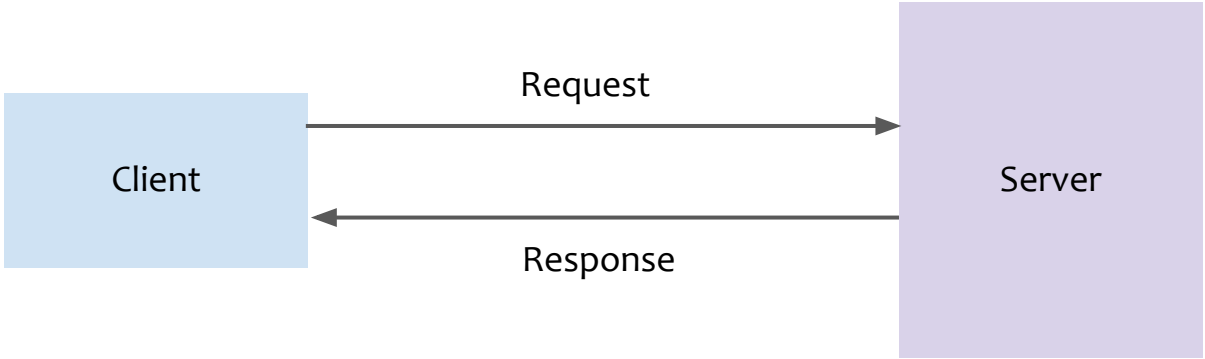
Remote Procedure Call (RPC)

Prof. Pramod Bhatotia
Chair of Decentralized Systems Engineering (DSE)
Department of Computer Science
https://dse.in.tum.de/

WS 2022/23

Distributed Systems 101:
A client-server application

A client-server application



The 8 Fallacies of distributed systems

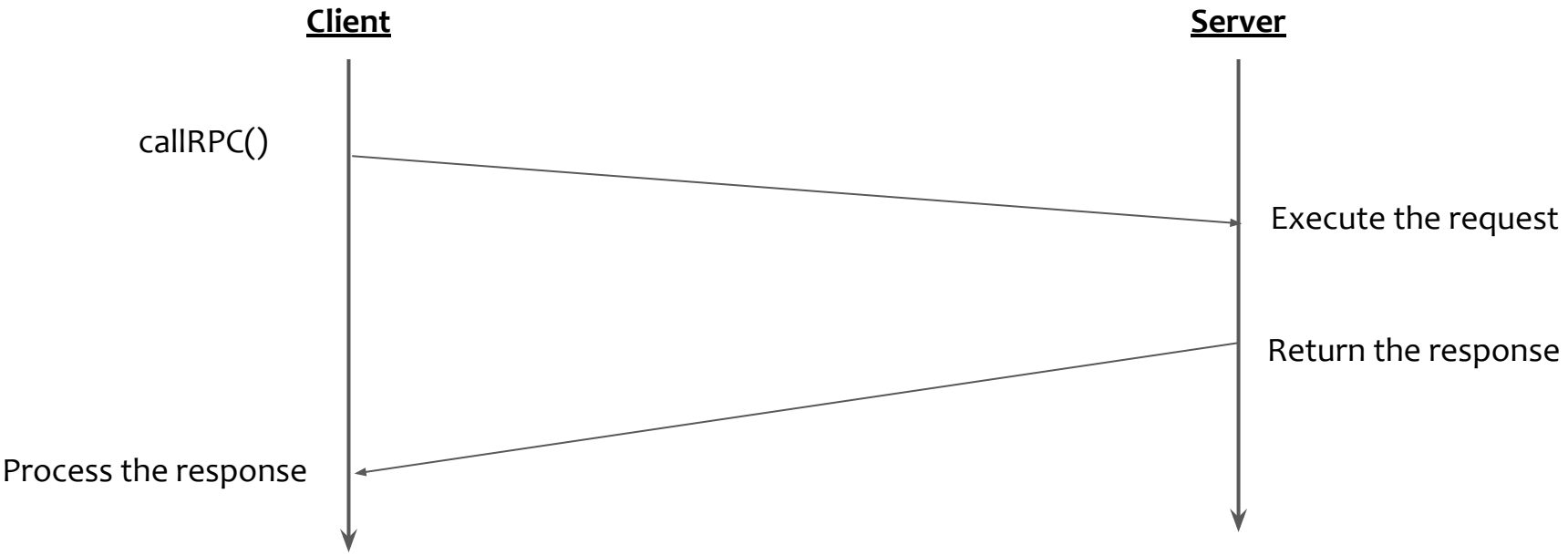
1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogenous

Remote procedure calls (RPCs)

- Extends the notion of local procedure calls
  - Allow two process to communicate (local or remote via network)
  - Residing in different address spaces
  - Presumes the existence of a low-level transport protocol (TCP/IP or UDP)

- How does a RPC work?
  - A client invokes an RPC, similar to a function call
  - Arguments are passed to the remote procedure
  - The caller waits for a response to be returned

RPC in action



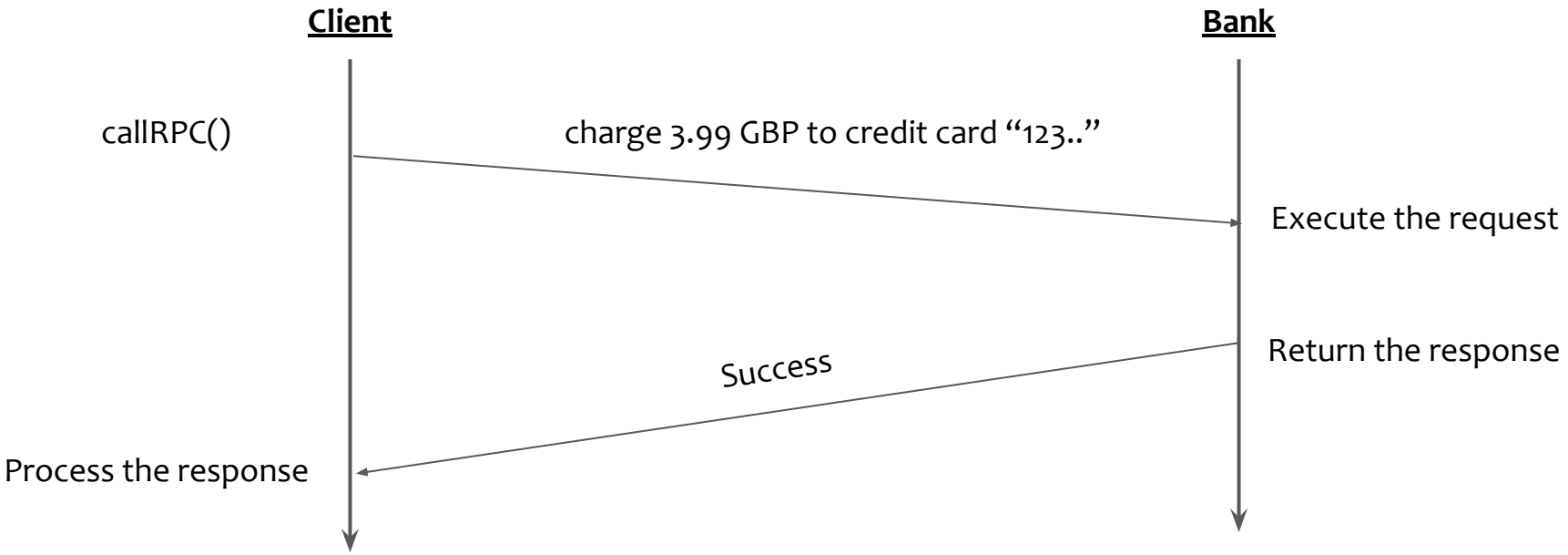
An application programmer:

1. Develops the client program
2. Develops the server program
3. Specifies the protocol for client-server communication

gRPC

- A high-performance and widely-deployed RPC framework
- Platform independent
- Applicable in almost all distributed computing use-cases:
  - Devices, mobile applications and browsers to backend cloud/data-center services.

How gRPC works?



A shopping cart example

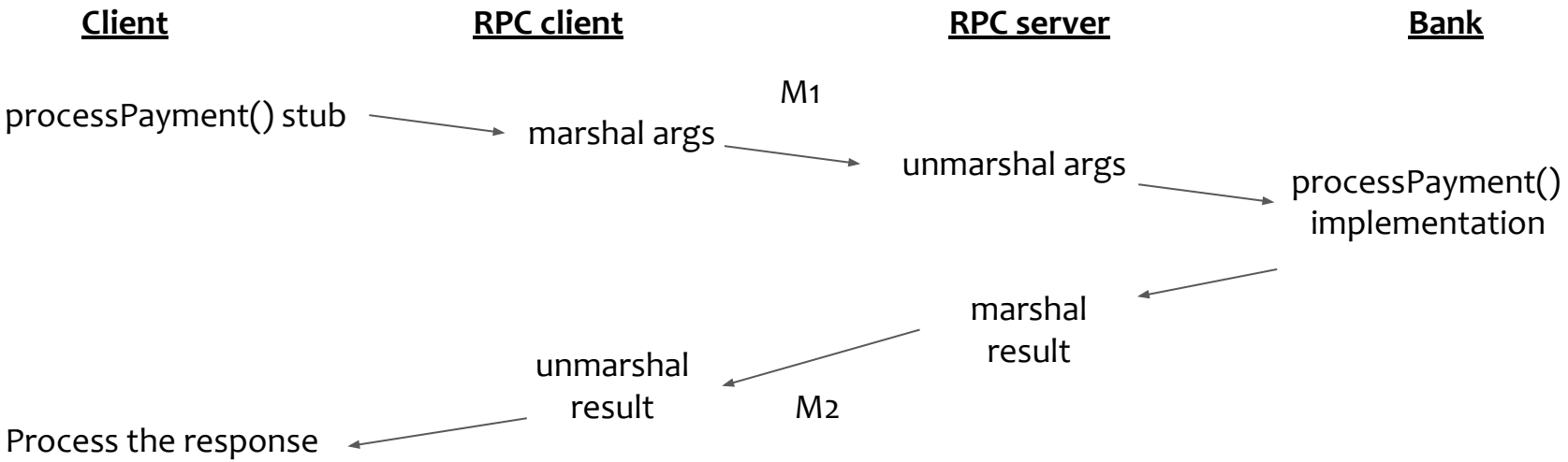
```
// Online shop handling customer's card details

Card card = new Card();
card.setCardNumber("123…");
card.setExpiryDate("10/2024");
card.setCVC("123");

Result result = paymentsService.processPayment(card, 3.99, Currency.GBP);


if (result.isSuccess()) {
        fulfilOrder();
}
```

gRPC in action



$m_1 =$

```
{
  "request": "processPayment",
  "card": {
    "number": "1234567887654321",
    "expiryDate": "10/2024",
    "CVC": "123"
  },
  "amount": 3.99,
  "currency": "GBP"
}
```

$m_2 =$

```
{
  "result": "success",
  "id": "XP61hHw2Rvo"
}
```

Data communication format

- The RPC framework needs to convert datatypes such that the caller's arguments are understood by the code being called, and vica-versa for the return values

- *Interface Definition Language (IDL)* provides language-independent type signatures of the functions that are being made available over RPC

- From the IDL, software developers can then automatically generate marshalling/unmarshalling code and RPC stubs for the respective programming languages of each service and its clients

Google's protocol buffers

- Protocol buffers provide a serialization format for packets of typed, structured data

- Specify the message format in a language-neutral, platform-neutral, extensible format for serialization/deserialization structured data (.proto files)

- The proto compiler is invoked on .proto files to generate code in various programming languages to manipulate the corresponding protocol buffer, i.e., to serialize and parse the whole structure to and from raw bytes

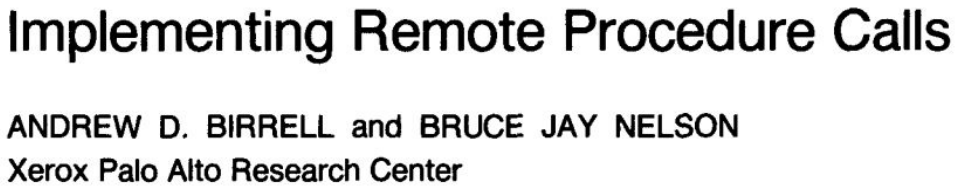
```
message Person {
  optional string name = 1;
  optional int32 id = 2;
  optional string email = 3;
}
```

Common error conditions

- Network data loss resulting in retransmission
  - Incorrect operations when the data is received multiple times
- Server process crashes during RPC operation
  - Before completing its task → Client retries the request
  - After completing its task and before responding → Recovery for a consistent state
- Client process crashes before receiving response
  - Client is restarted, server discards response data

References

- gRPC: https://developers.google.com/protocol-buffers
- Protocol buffers: https://developers.google.com/protocol-buffers
- Implementing Remote Procedure Calls
  - https://web.eecs.umich.edu/~mosharaf/Readings/RPC.pdf



Implementing Remote Procedure Calls

ANDREW D. BIRRELL and BRUCE JAY NELSON
Xerox Palo Alto Research Center