Hochschule RheinMain

REMOTE PROCEDURE CALLS

Verteilte Systeme

Prof. Dr. Georg Hinkel
10.05.2024

GLIEDERUNG

Hochschule RheinMain

Datum	Vorlesung	Übungsblatt	Abgabe
19.04.2024	Einführung	HamsterLib	06.05.2024
26.04.2024	Netzwerkprogrammierung	Theorie	
03.05.2024	World Wide Web	HamsterRPC 1	20.05.2024
10.05.2024	Remote Procedure Calls	Theorie	
17.05.2024	Webservices	HamsterRPC 2	03.06.2024
24.05.2024	Fehlertolerante Systeme	Theorie	
31.05.2024	Transportsicherheit	HamsterREST	17.06.2024
07.06.2024	Architekturen für Verteilte Systeme	Theorie	
14.06.2024	Internet der Dinge	HamsterIoT	01.07.2024
21.06.2024	Namen- und Verzeichnisdienste	Theorie	
28.06.2024	Authentifikation im Web	HamsterAuth	15.07.2024
05.07.2024	Infrastruktur für Verteilte Systeme	Theorie	
12.07.2024	Wrap-Up	HamsterCluster (Bonus)	16.08.2024

Hochschule RheinMain

Agenda

- Remote Procedure Calls
  - Grundprinzip
  - Netzdatenrepräsentationen
  - Semantik im Fehlerfall
- gRPC

Lernziele

- RPCs anwenden können
- Überblick über verbreitete Technologien haben

MOTIVATION
Verschiedene Welten

Hochschule RheinMain

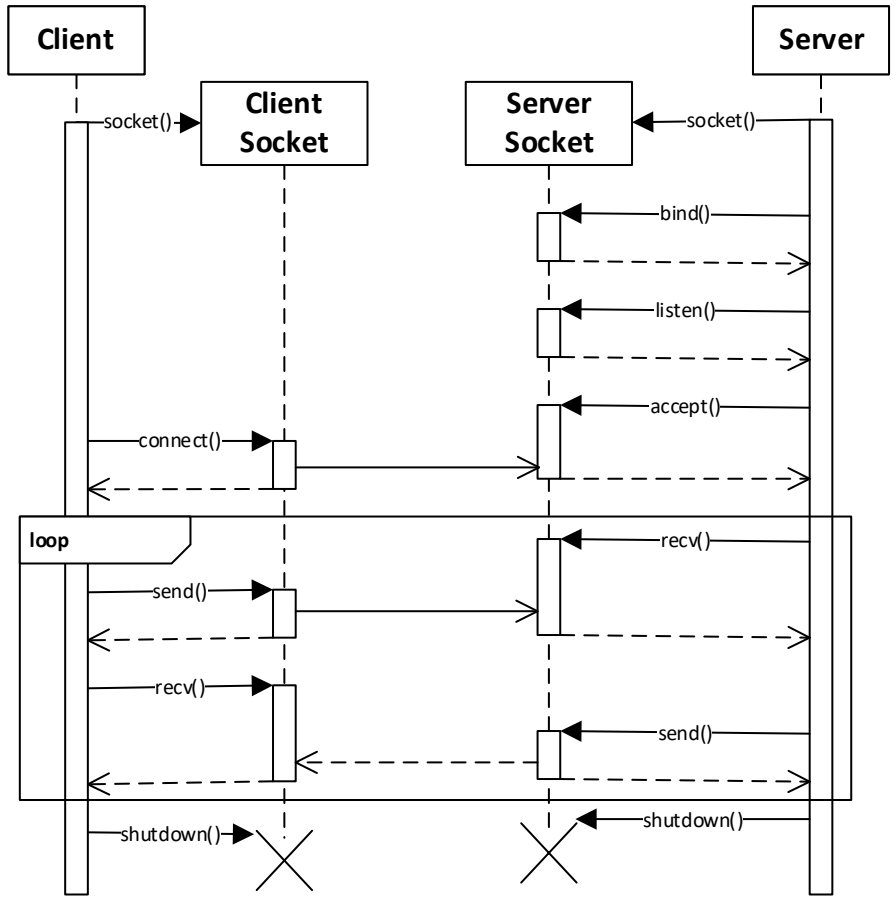
```
public interface Calculator {
   String Calculate(String expression);
}
```


```
01010111011010000110000101111010000100
00001100100011011110010000011110010 1
10111101110101001000001100111011001 0
10111010000100000011010010110011000 10
00000111100101101111011110101001000000
11011010111010101101100011101000110 10
01011100000110110001111001001000000 11
10011011010010111100001000000110001 0
01111001001000000110111001101001011 01
11001100101001111 11…
```

- Kommunikation lokal: Methodenaufruf
  - Von der Programmiersprache/Compiler bereitgestellte Abstraktion, um Code aufzurufen
  - Übergabe von Parametern, Rücksprungadresse
  - Rückgabe von Rückgabewerten
- Kommunikation TCP/UDP: Bytestrom/Paket
  - Speicheradressen auf entferntem Host nicht gültig (Code, Daten)
  - Potentiell fehleranfälliges Netzwerk

MOTIVATION
Aufrufsemantik

Hochschule RheinMain



- Drastische Unterschied in der Aufrufsemantik
  - Methodenaufruf: Aufgerufene Methode wird erst aktiv, wenn Methode aufgerufen wird
  - Sockets: Server muss aktiv warten

MOTIVATION
Was ist zu tun?

Hochschule RheinMain

```
public interface Calculator {
   String Calculate(String expression);
}
```


```
01010111011010000110000101111010000100
00001100100011011110010000001111001101
10111101110101001000000110011101100010
10111010000100000011010010110011000010
00000111100101101111011110101001000000
11011010111010101101100011101000110100
01011100000110110001111001001000000011
10011011010010111100000100000011000010
01111001001000000110111001101001011010
11001100101001111111…
```

Aufrufprotokoll

- Wie schickt der Client Nachrichten an den Server?
- Wie geht der Server mit eingehenden Nachrichten um?

Nachrichtensemantik für den Client

- Auswahl der Methode
- Auswahl des Objektes (?)
- Übergebene Parameter

Nachrichtensemantik für den Server

- Rückgabewerte
- Fehlerinformationen

REMOTE PROCEDURE CALLS (RPC)

Deutsch: Fernaufruf, ungebräuchlich

Hochschule **RheinMain**

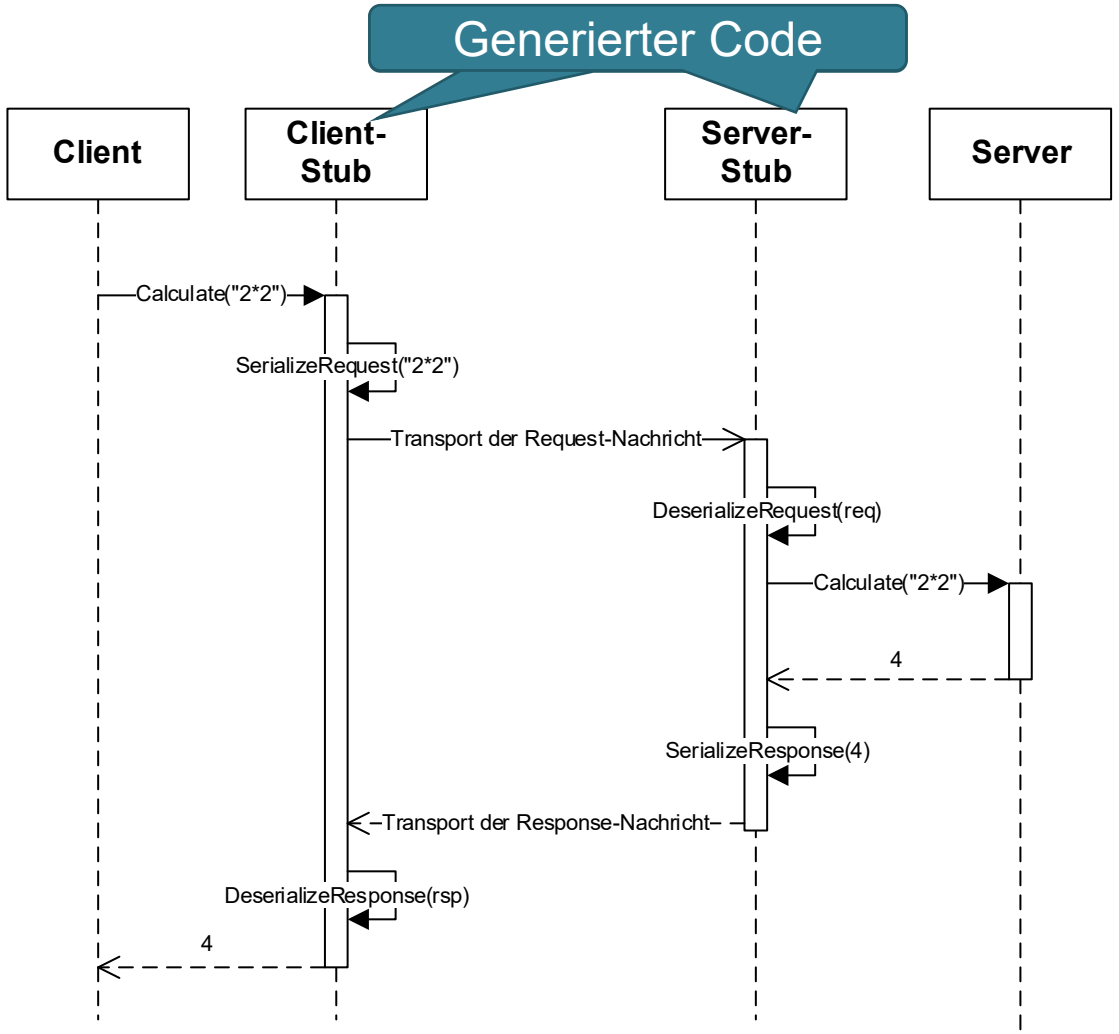
- Idee: Standardisiere Nachrichtensemantik für Aufruf entfernter Methode
  - Dissertation Nelson (1981, XPARC)
  - These: Vereinfacht Konstruktion verteilter Systeme

- RPC Framework
  - Definiert Standard, wie die versendeten/empfangenen Nachrichten aussehen müssen
  - Definiert Methodenauswahl (optional, sonst nur eine Methode pro Verbindung)
  - Ggf. zusätzliche Metainformationen (Header)
  - Idealerweise **programmiersprachenunabhängig**
  - Beispiele: SunRPC, Apache Thrift, gRPC

- Unterstützung durch Tools
  - Bestandteil des RPC Framework
  - Üblicherweise domänenspezifische Sprache für Schnittstellendefinition
  - Üblicherweise Code-Generierung für Serialisierung, Stümmel (stub)

REMOTE PROCEDURE CALLS

Ablauf



- RPC generiert Stümmel (stub) für Client und Server
  - Bilden Schnittstellendefinition ab
  - Implementieren Serialisierung und Deserialisierung
  - Implementieren Übermittlung der Nachricht
- Client verwendet typischerweise Stümmel direkt
  - Übergabe der Serveradresse oder des Kanals per Konstruktor
- Implementierung der Server-Funktionalität häufig mittels Inheritance
  - Stub bietet virtuelle/abstrakte Methoden

REMOTE PROCEDURE CALLS
Sicherheit

Hochschule **RheinMain**

- Probleme
  - Authentifizierung: Ist das wirklich der richtige Server?
  - Autorisierung: Darf der Client die Methode ausführen?
  - Verschlüsselung

- Ausführliche Betrachtung in späterer Vorlesung

REMOTE PROCEDURE CALLS
Netzdatendarstellung

Hochschule RheinMain

- Grundproblem: Daten müssen in Bytestrom umgewandelt werden
- Lösung: Generische Serialisierung
  - Wie das geht haben Sie in Programmiermethoden gelernt…
- Aber: Speicheradressen gehen bei Serialisierung/Deserialisierung verloren
  - Globaler Adressraum, falls vorhanden
  - Ersetzen bekannter Objekte durch Marker / Identifier, Rekonstruktion auf Empfängerseite

REMOTE PROCEDURE CALLS
Netzdatendarstellungen

Hochschule RheinMain

- XML (eXtensible Markup Language)
  - Textbasiertes Austauschformat
  - Standardisiert von W3C
  - Parser in sehr vielen Programmiersprachen verfügbar
  - Verbreitetes Format, um Struktur vorzugeben (XML Schema, eigener Standard)
  - Schachtelung durch Unterelemente, Vererbung
  - Selbstbeschreibende Struktur, einfache Versionierung
  - Noch lesbar für Menschen
  - Nachteil: Verhältnismäßig platzintensiv wegen Redundanz von Headern

```
<Person
  firstName="Georg"
  lastName="Hinkel">
  <Studiengang>AI</Studiengang>
</Person>
```

Marker auf komplexes Objekt

REMOTE PROCEDURE CALLS
Netzdatendarstellungen

Hochschule RheinMain

- JSON (JavaScript Object Notation)
  - Schlankes, textbasiertes Austauschformat
  - RFC 7159, abgeleitet von ECMAScript
  - Parser in sehr vielen Programmiersprachen verfügbar, aber Schemata kaum verbreitet
  - Schachtelung durch Listen oder Objekte ➔ JavaScript-Notation für Array-Listen und Hash-Tabellen
  - Selbstbeschreibende Struktur, einfache Versionierung
  - Noch lesbar für Menschen
  - Nachteil: Immer noch verhältnismäßig platzintensiv wegen Redundanz von Headern, aber effizienter als XML

```
{
  “firstName“: “Georg“,
  “lastName“: “Hinkel“,
  “studiengang“: “AI“
}
```

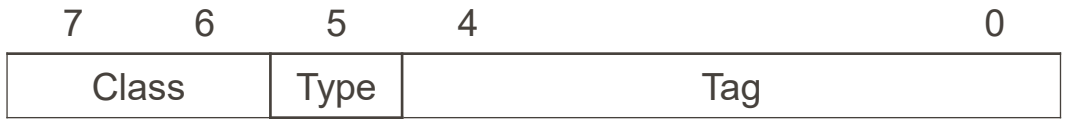
Hochschule RheinMain

- ASN.1 (ISO Abstract Syntax Notation Number 1)
  - Binärformat, explizite Typisierung der übertragenen Daten
  - Häufig verwendet für Telekommunikation: CANopen, LDAP, UMTS/LTE, VoIP, …
  - Standard-Repräsentation: Typ, Länge, Inhalt
  - Schachtelung möglich (Sequenz, Choice, Set typisiert/untypisiert)
  - Nachteil: Client und Server müssen gleiche Version referenzieren

count ::= INTEGER

0 2	Identifier : (Integer, primitive, universal)
0 1	Länge : 1 Byte
2 A	Inhalt : 42

1 Byte für Länge falls < 128
Sonst erstes Byte um Anzahl Längenbytes anzugeben

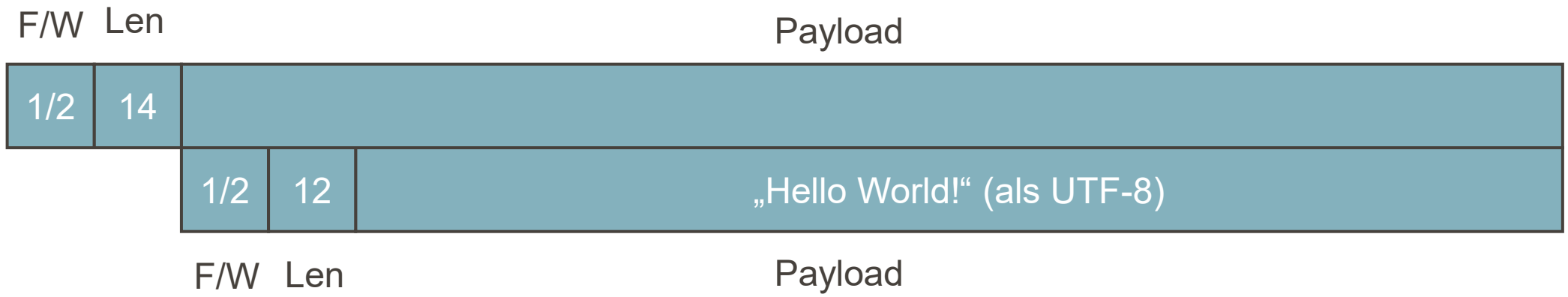


Tag	1	Boolean
	2	Integer
	…	…
	16	Sequence
Type	0	Primitive
	1	Constructed
Class	00	Universal
	01	Application

REMOTE PROCEDURE CALLS
Netzdatendarstellung

Hochschule **RheinMain**

- Protocol Buffers (ProtoBuf)
  - Platzeffizientes Binäres Austauschformat, explizite Typinformationen
  - Parser in vielen Programmiersprachen verfügbar
  - Schachtelung durch Repetitions, Objekte, zusätzlich Enumerationen und One-Of
  - Kaum selbstbeschreibend, aber Feldnummern ➔ ab-/aufwärtskompatibel



HelloReply besteht nur aus 16 byte
- Erstes Byte: Feldnummer << 3 + Wiretype
- Nächste Bytes sind Länge (1 Byte für < 128)
- Nachfolgend Payload

```
message HelloReply {
  string message = 1;
}
```

INTEGER VARIABLER LÄNGE

Hochschule RheinMain

- Lösung in ProtocolBuffers
  1. Konvertiere Zahl nach Binär
  2. Gruppiere in Blöcken von 7 Bits, auffüllen mit 0
  3. Blöcke umdrehen
  4. Erstes Bit ist 1, wenn weitere Gruppe folgt, sonst 0
- Beispiel: 2023
  - 0111 1110 0111
  - 0001111 1100111
  - 1100111 0001111
  - 11100111 00001111 ➔ 2 Bytes
- Vorteil: Effizient invertierbar, platzsparend
  - Längenfeld kann bei numerischen Werten entfallen
  - Häufige Zahlen (Längen) < 128 als ein Byte
  - Parser weiß anhand des ersten Bits, ob weitere Bits noch zur Zahl gehören
  - Addition weiterer Bit-Blöcke durch einfache Bit-Shifts

REMOTE PROCEDURE CALLS
Transportschicht

Hochschule RheinMain

- Transport der Nachrichten: traditionell über TCP oder UDP
  - Auswahl nach Anforderung an die Dienstgüte
- Problem: Weiterer offener Port
  - Firewall muss passend konfiguriert sein, um Verbindungsversuch zuzulassen
- Problem 2: Keine nebenläufigen Anfragen auf derselben Verbindung
  - Socket weiterhin in Verwendung ➔ keine Nebenläufigkeitstransparenz
- Lösungsidee: Verwende HTTP als Transportschicht
  - HTTP Header als zusätzliche Metainformationen
  - HTTP Port(s) als einzige offenen Ports
  - Ab HTTP/2.0 auch Nebenläufigkeitstransparenz

REMOTE PROCEDURE CALLS
Fehlerquellen

Hochschule RheinMain

- Lokaler Funktionsaufruf: Exception

- RPCs: Mehrere Fehlerquellen
  - Nachrichtenverlust oder Verzögerung der Anfragenachricht
  - Eigentlicher Aufruf beim Server
  - Nachrichtenverlust oder Verzögerung der Antwortnachricht

REMOTE PROCEDURE CALLS
Fehlerfall

Hochschule **RheinMain**

- Unterschiedliche Semantiken im Fehlerfall
  - At-least-once: Aufruf beim Server findet mindestens einmal statt
  - At-most-once: Aufruf beim Server findet höchstens einmal statt
  - Exactly-once: Aufruf beim Server findet genau einmal statt

- Semantik im Fehlerfall häufig vom Transportprotokoll „durchgereicht“
  - Zuverlässiges Transportprotokoll (e.g., TCP) ➔ at-least-once, häufigster Fall
  - Unzuverlässiges Transportprotokoll (e.g., UDP) ➔ at-most-once

REMOTE PROCEDURE CALLS
Fehlerfall

Hochschule RheinMain

- „Orphan“-Problem
  - Netzwerkverbindung bricht ab, nachdem der RPC abgesetzt ist, aber vor Response
  - Unklar, ob RPC überhaupt ausgeführt wurde
  - RPC kann weitere Aktivitäten nach sich ziehen, aber niemand wartet mehr darauf
  - Ggf. nach Neustart Eintreffen von Antworten aus „früherem Leben“

REMOTE PROCEDURE CALLS
Beispiele

Hochschule **RheinMain**

- SunRPC / Open Network Computing (ONC) RPC
  - Verwendet für sehr hardwarenahe Programmierung
  - Spracheinbettung in C
  - Verschiedene Transportschichten
    - TCP, UDP
    - Dienstgüte abhängig vom Transportprotokoll
- Beliebtes RPC-Protokoll in Vorgängerveranstaltung

```
const MAX_FILENAME_LEN = 255;
typedef string t_filename <MAX_FILENAME_LEN >;
const MAX_CONTENT_LEN = 255;
typedef string t_content <MAX_CONTENT_LEN >;

struct s_filewrite {
    t_filename filename;
    t_content content;
};
struct s_chmod {
    t_filename filename;
    long mods;
};
…
program fileservice {
    version fsrv {
        int fsrv_mkdir(string) = 1;
        int fsrv_rmdir(string) = 2;
        int fsrv_chdir(string) = 3;
        int fsrv_writefile(s_filewrite) = 4;
        string fsrv_readfile(string) = 5;
        s_fstat fsrv_fileattr(string) = 6;
        int fsrv_chmod(s_chmod) = 7;
    } = 1;
} = 0x30000001;
```

REMOTE PROCEDURE CALLS
Beispiele

Hochschule RheinMain

- Idee: Transparenter Zugriff auf entfernte Objekte durch Registry
  - Objektorientierung auch bei verteilten Systemen anstatt prozeduraler Schnittstelle
  - Objekte werden von zentraler Registry verwaltet, um auch Ortsunabhängigkeit sicher zu stellen

- Java Remote Method Invocation (RMI)
  - Kommunikationsprotokoll, RPC-Protokoll und Klassenbibliothek
  - Remote Object und Remote Reference als Abstraktion von entfernten Objekten

- Ablauf
  1. Server registriert Remote Object bei Registry
  2. Client schlägt in Registry nach und bekommt Referenz auf entferntes Objekt
  3. Client ruft Methode auf, Parameter werden übertragen. Falls notwendig, wird die Klasse der übertragenen Objekte mitübertragen
  4. Server führt Methode aus. Dafür müssen ggf. Klassen vom Client **ggf. Fremdcode** nachgeladen werden

REMOTE PROCEDURE CALLS
Beispiele

Hochschule **RheinMain**

- Windows Communication Foundation (WCF)
  - Bestandteil .NET Framework ab 3.0 (2006)
  - Verschiedene Protokolle
  - Trennung von Code und Protokollinformationen
  - Simultane Unterstützung mehrerer Protokolle
  - Limitiert auf Windows + .NET Framework
  - Kommt ohne Code-Generator aus
- Entwicklung eingestellt
  - Mangelnde Effizienz
  - Keine Plattformunabhängigkeit

```
[ServiceContract]
class HelloService
{
  [OperationContract]
  [PrincipalPermission(SecurityAction.Demand,
    Role = “Adminstrators")]
  [TransactionFlow(TransactionFlowOption.Mandatory)]
  [OperationBehavior(TransactionScopeRequired = true,
    TransactionAutoComplete = true)]
  String Hello(String Greeting) {return Greeting;}
}
```

```
<bindings>
  <wsHttpBinding>
    <binding name="Binding1“  transactionFlow="true">
      <security mode="Message">
        <message clientCredentialType="Windows“/>
      </security>
      <reliableSession enabled="true" />
    </binding>
  </wsHttpBinding>
</bindings>
```

REMOTE PROCEDURE CALLS
Beispiele

Hochschule RheinMain

- JSON-RPC
  - Aktuelle Version 2.0 (2010)
  - Schnittstellenspezifikation in TypeScript
  - Unterstützung für Web
  - Bibliotheken in anderen Programmiersprachen
  - Unterstützung verschiedener Transportprotokolle, u.a. Websockets
- Beispiel: LSP (Language Server Protocol)
  - Erweitert JSON-RPC um Content-Length Header

```
{
   "jsonrpc": "2.0",
    "method": "subtract",
    "params": {
        "minuend": 42,
        "subtrahend": 23
    },
    "id": 3
}
```

```
{"jsonrpc": "2.0", "result": 19, "id": 3}
```

REMOTE PROCEDURE CALLS
Beispiele

Hochschule **RheinMain**

- gRPC
  - Ursprung Google, veröffentlicht 2015
  - Version programmiersprachenabhängig
  - Unterstützung von vielen Programmiersprachen
  - Transportprotokoll: HTTP/2.0 oder höher
  - Einfache Proto-Sprache
  - Nachrichtenformat ProtoBuf ➔ binär
  - Parallelitätstransparent
  - Hohe Verbreitung
- Zusätzliche Metadaten (Header)
- Zusätzliche Methodentypen
  - Ermöglicht durch HTTP/2.0 Server Streaming

gRPC

```
syntax = "proto3";

service Greeter {
  rpc SayHello (HelloRequest) returns (HelloReply);
}

message HelloRequest {
  string name = 1;
}

message HelloReply {
  string message = 1;
}
```

METHODENTYPEN IN GRPC

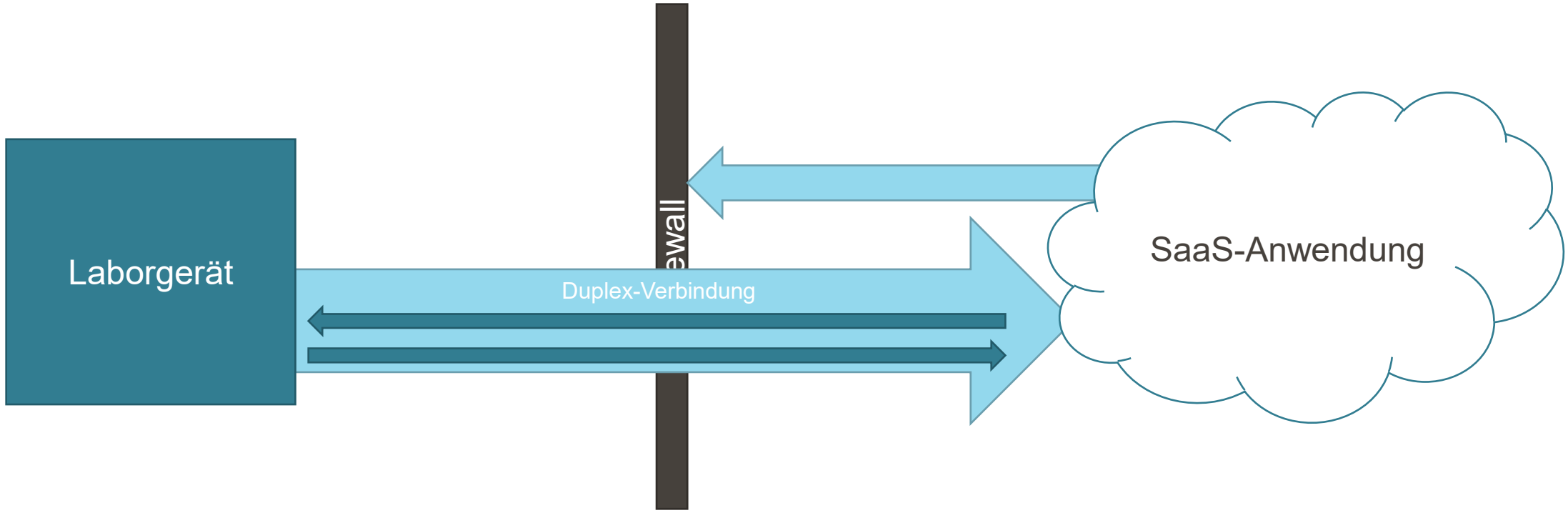
Hochschule RheinMain

- Unary
  - Klassisches Request/Response ➔ Eine Frage, eine Antwort
- Client-Streaming
  - Client darf beliebig viele Nachrichten schicken, Antwort des Servers beendet RPC-Aufruf
- Server-Streaming
  - Server kann beliebig viele Nachrichten schicken, zeitlich ggf. verzögert ➔ Subscription
  - Client kann RPC bei Bedarf abbrechen
- Duplex
  - Client und Server dürfen jederzeit Nachrichten schicken und RPC abbrechen

BEISPIEL: DUPLEX FÜR ROLLENTAUSCH

Server-initiated Connections in SiLA2 (Laborautomatisierung)

Hochschule **RheinMain**



- Duplexverbindungen werden genutzt um Firewall-Probleme zu umgehen

ZUSAMMENFASSUNG
Remote Procedure Calls

Hochschule RheinMain

- Remote Procedure Calls schaffen Zugriffstransparenz
  - Aufruf von Funktionen auf entferntem Rechner so als ob lokal
- Wichtige Elemente
  - Schnittstellenbeschreibungssprache (IDL)
  - Werkzeuge für Code-Generierung, falls erforderlich
- Neue Fehlerquellen
  - Verschiedene Semantiken im Fehlerfall
  - Orphan-Problem
- gRPC
  - RPC Protokoll auf Basis von HTTP/2.0
  - Zusätzlich Nebenläufigkeitstransparenz
  - Reichhaltigere Semantik durch mehr Methodentypen

MÖGLICHE PRÜFUNGSAUFGABEN

Hochschule RheinMain

- Diskutieren Sie, inwieweit RPCs sich eignen, um eine Zugriffstransparenz zu erreichen!
- Warum bedeuten die verwendeten Stümmel keinen zusätzlichen Entwicklungsaufwand?
- Nennen Sie Beispiele für Netzdatendarstellungen!
- Nennen Sie Beispiele für RPC-Frameworks!
- Berechnen Sie die Bitdarstellung für eine gegebene Zahl in Protocol Buffer!
- Erläutern Sie das „Orphan“-Problem!
- Wie kann man Parallelitätstransparenz in RPCs erreichen?
- Erläutern Sie die Methodentypen in gRPC!