

Cloud Computing

Kapitel 1: Kommunikationssysteme im Internet

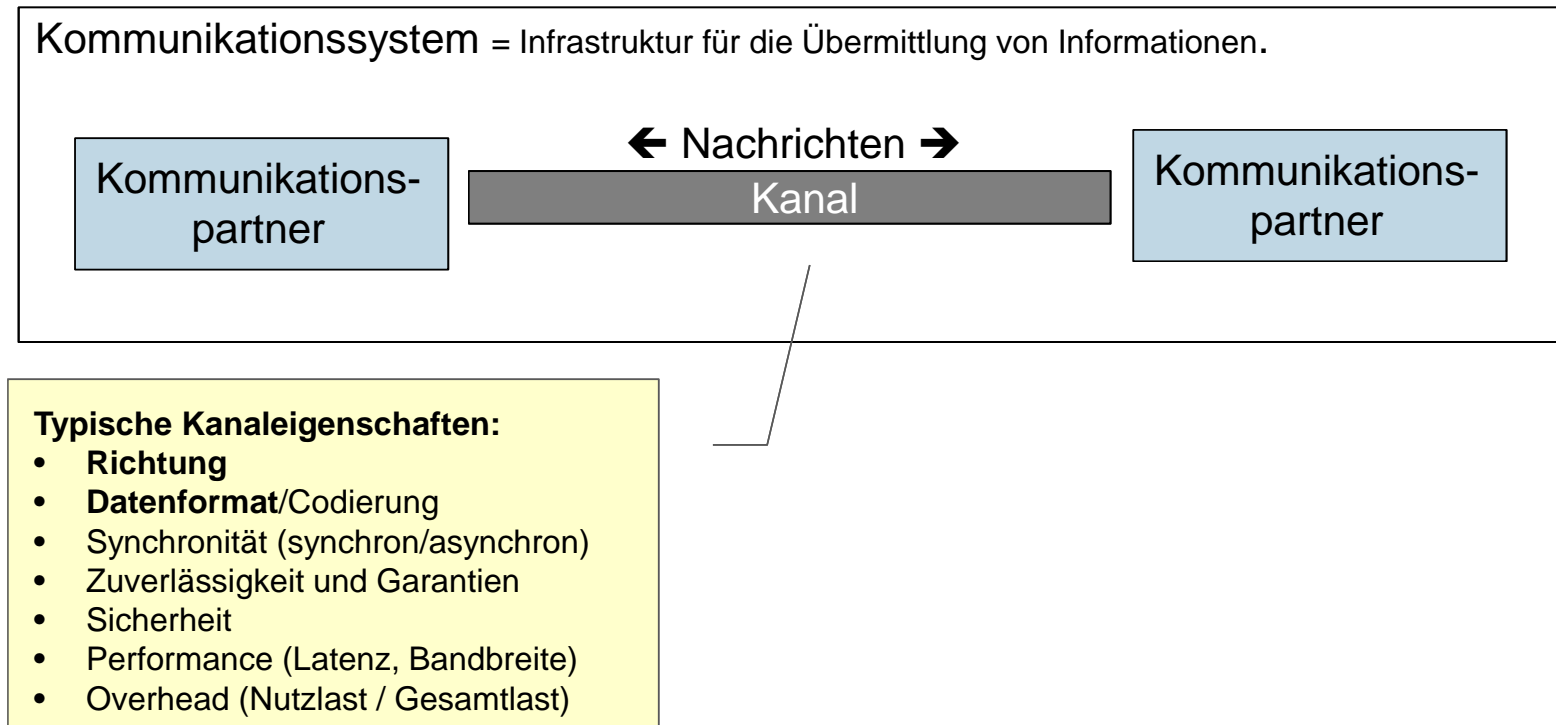
Dr. Josef Adersberger

Grundlagen von Kommunikationssystemen im Internet

„Man kann nicht nicht kommunizieren!“
Paul Watzlawik

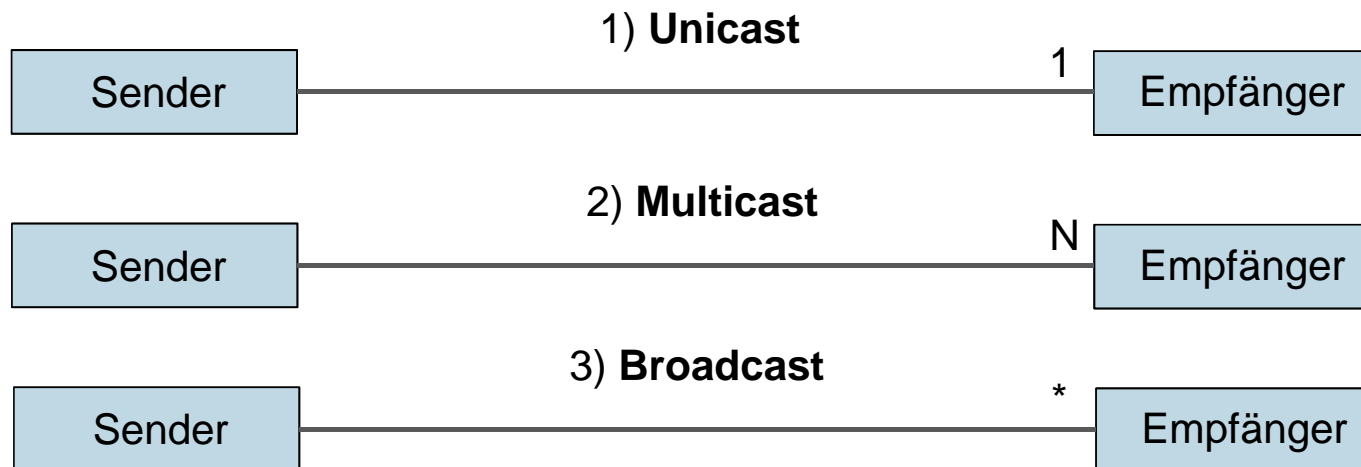
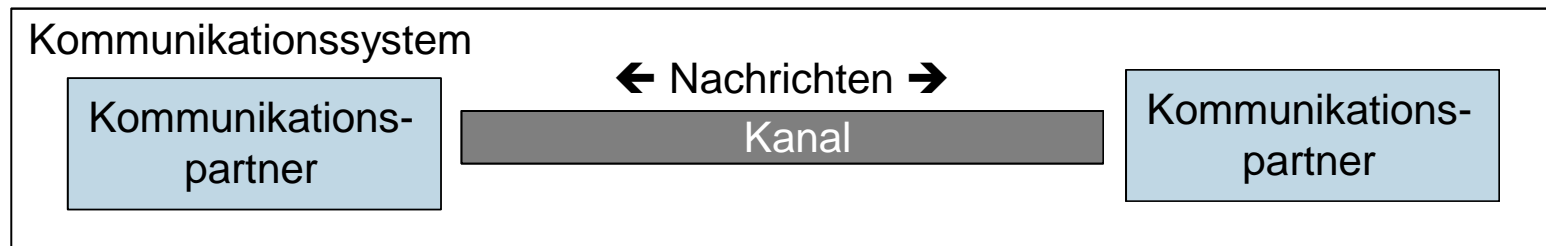


Ein allgemeines Kommunikationsmodell im Internet. Angelehnt an das Modell von Shannon/Weaver.

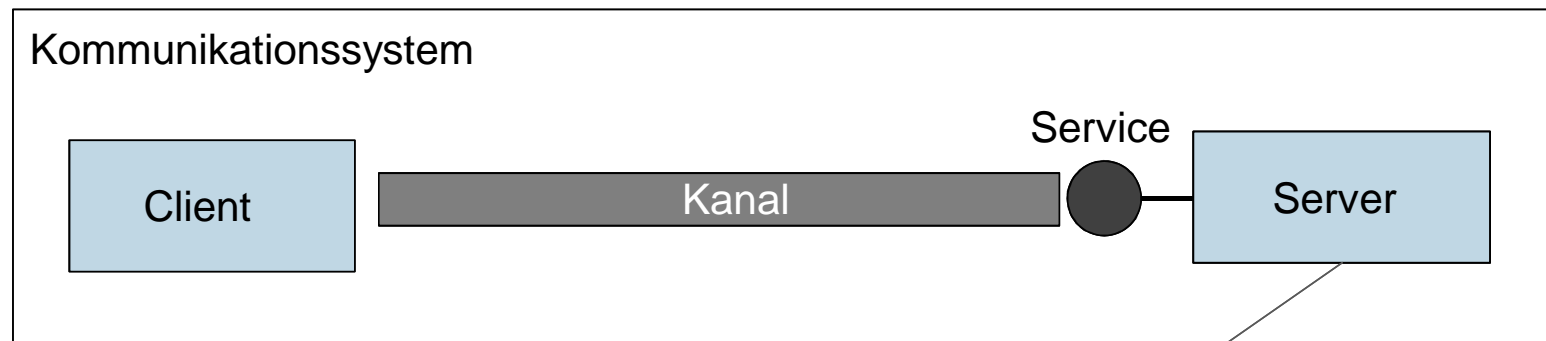


Klassifikation von Kommunikationssystemen:

(A) Kardinalität der Empfänger einer Nachricht.



Das Client/Server- Kommunikationsmodell ist im Internet weit verbreitet.

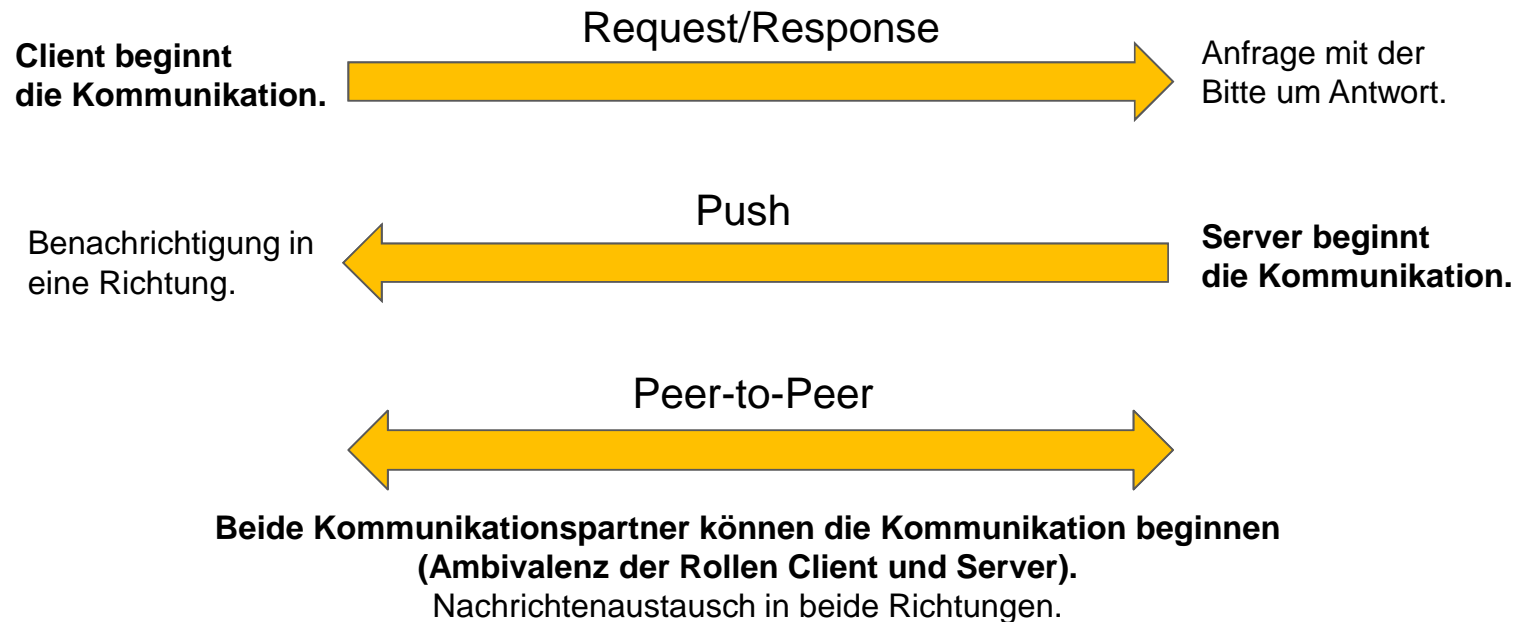
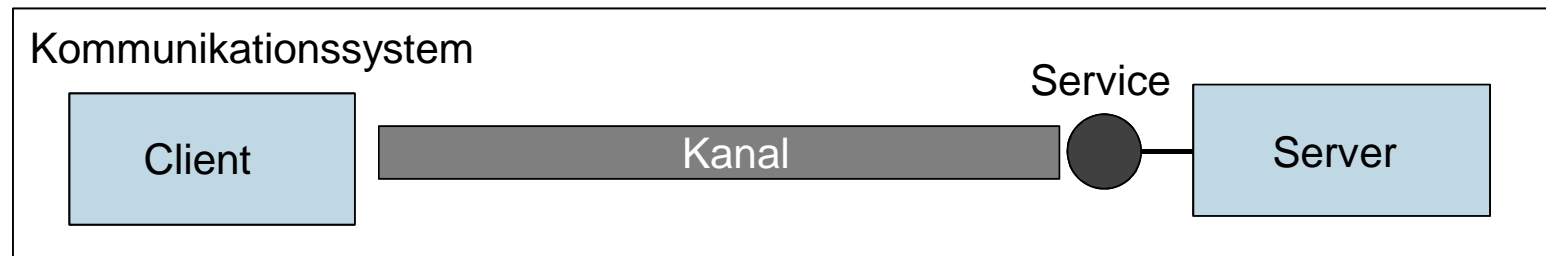


Ein **Service** ist eine Funktionalität, die über eine definierte Schnittstelle zur Verfügung stellt. Jeder Service ist definiert durch eine **Serviceschnittstelle**.

Eine **Serviceschnittstelle** ist ein Vertrag zwischen Nutzer und Anbieter über Syntax und Semantik der Service-Nutzung und enthält optional Zusicherungen in Hinblick auf den **Quality of Service**.

Klassifikation von Kommunikationssystemen:

(B) Wer beginnt mit der Kommunikation?



Es sind eine Reihe an Kommunikationsprotokollen für das Internet standardisiert.

OSI-Schicht *	TCP/IP-Schicht	Beispiel
Anwendungen (7)	Anwendungen	HTTP, FTP, SMTP, POP, Telnet, OPC UA
Darstellung (6)		
Sitzung (5)		
		SOCKS
Transport (4)	Transport	TCP, UDP, SCTP
Vermittlung (3)	Internet	IP (IPv4, IPv6)
Sicherung (2)	Netzzugang	Ethernet, Token Bus, Token Ring, FDDI, IPoAC
Bitübertragung (1)		

*) Entsprechend dem OSI-Referenzmodell für Kommunikationsprotokolle.

Quelle: <http://de.wikipedia.org>

Basis aller folgend betrachteten Kommunikationstechnologien ist TCP und teilweise HTTP.

TCP (Transmission Control Protocol)	
Familie:	Internetprotokollfamilie
Einsatzgebiet:	Zuverlässiger bidirektionaler Datentransport
TCP im TCP/IP-Protokollstapel:	
Anwendung	HTTP SMTP ...
Transport	TCP
Internet	IP (IPv4, IPv6)
Netzzugang	Ethernet Token Bus Token Ring FDDI ...
Standards:	RFC 793 ↗ (1981) RFC 1323 ↗ (1992)

- Ab 1973 entwickelt und 1981 standardisiert.
- Zuverlässige Voll-Duplex Ende-zu-Ende Verbindung.
- Ein Endpunkt ist eine IP + Port.

HTTP (Hypertext Transfer Protocol)	
Familie:	Internetprotokollfamilie
Einsatzgebiet:	Datenübertragung, Hypertext u. a.
Port:	80/TCP
HTTP im TCP/IP-Protokollstapel:	
Anwendung	HTTP
Transport	TCP
Internet	IP (IPv4, IPv6)
Netzzugang	Ethernet Token Bus Token Ring FDDI ...
Standards:	RFC 1945 ↗ (HTTP/1.0, 1996) RFC 2616 ↗ (HTTP/1.1, 1999)

- HTTP 1.0: 1989 am CERN entwickelt.
- HTTP 1.1: Connection Pooling / Keepalive, HTTP-Pipelining, Methoden PUT und DELETE.
- HTTP 2.0 (Juli 2013): Multiplexing, Verschlüsselung als Standard, div. Performance-Optimierungen, Push.

Ein Beispiel für eine HTTP-Kommunikation.

<http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html>

Request

```
GET /index.html HTTP/1.1
Host: www.oreilly.com
User-Agent: Mozilla/5.0
Accept: text/xml, text/html, application/xml
Accept-Language: us, en
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859, UTF-8
Keep-Alive: 300
Connection: Keep-Alive
```

Response

```
HTTP/1.1 200 OK
Date: Mon 26 Jul 2010 15:35:55 GMT
Server: Apache
Last-Modified: Fri 23 Jul 2010 14:01:13 GMT
Accept-Ranges: bytes
Content-Length: 43302
Content-Type: text/html
X-Cache: MISS from www.oreilly.com
Keep-Alive: timeout=15, max=1000
Connection: Keep-Alive

<!DOCTYPE html PUBLIC "...">
<html>...</html>
```

<http://www.ietf.org/rfc/rfc2046.txt?number=2046>

Referenz: HTTP Status-Codes

Informational Status Codes 100 — Continue [The server is ready to receive the rest of the request.] 101 — Switching Protocols [Client specifies that the server should use a certain protocol and the server will give this response when it is ready to switch.]	Client Request Incomplete 400 — Bad Request [The server detected a syntax error in the client's request.] 401 — Unauthorized [The request requires user authentication. The server sends the WWW-Authenticate header to indicate the authentication type and realm for the requested resource.] 402 — Payment Required [reserved for future.] 403 — Forbidden [Access to the requested resource is forbidden. The request should not be repeated by the client.] 404 — Not Found [The requested document does not exist on the server.] 405 — Method Not Allowed [The request method used by the client is unacceptable. The server sends the Allow header stating what methods are acceptable to access the requested resource.] 406 — Not Acceptable [The requested resource is not available in a format that the client can accept, based on the accept headers received by the server. If the request was not a HEAD request, the server can send Content-Language, Content-Encoding and Content-Type headers to indicate which formats are available.] 407 — Proxy Authentication Required [Unauthorized access request to a proxy server. The client must first authenticate itself with the proxy. The server sends the Proxy-Authenticate header indicating the authentication scheme and realm for the requested resource.] 408 — Request Time-Out [The client has failed to complete its request within the request timeout period used by the server. However, the client can re-request.] 409 — Conflict [The client request conflicts with another request. The server can add information about the type of conflict along with the status code.] 410 — Gone [The requested resource is permanently gone from the server.] 411 — Length Required [The client must supply a Content-Length header in its request.] 412 — Precondition Failed [When a client sends a request with one or more If... headers, the server uses this code to indicate that one or more of the conditions specified in these headers is FALSE.] 413 — Request Entity Too Large [The server refuses to process the request because its message body is too large. The server can close connection to stop the client from continuing the request.] 414 — Request-URI Too Long [The server refuses to process the request, because the specified URI is too long.] 415 — Unsupported Media Type [The server refuses to process the request, because it does not support the message body's format.] 417 — Expectation Failed [The server failed to meet the requirements of the Expect request-header.]	Server Errors 500 — Internal Server Error [A server configuration setting or an external program has caused an error.] 501 — Not Implemented [The server does not support the functionality required to fulfill the request.] 502 — Bad Gateway [The server encountered an invalid response from an upstream server or proxy.] 503 — Service Unavailable [The service is temporarily unavailable. The server can send a Retry-After header to indicate when the service may become available again.] 504 — Gateway Time-Out [The gateway or proxy has timed out.] 505 — HTTP Version Not Supported [The version of HTTP used by the client is not supported.]
Client Request Successful 200 — OK [Success! This is what you want.] 201 — Created [Successfully created the URI specified by the client.] 202 — Accepted [Accepted for processing but the server has not finished processing it.] 203 — Non-Authoritative Information [Information in the response header did not originate from this server. Copied from another server.] 204 — No Content [Request is complete without any information being sent back in the response.] 205 — Reset Content [Client should reset the current document. I.e. A form with existing values.] 206 — Partial Content [Server has fulfilled the partial GET request for the resource. In response to a Range request from the client. Or if someone hits stop.]		
Request Redirected 300 — Multiple Choices [Requested resource corresponds to a set of documents. Server sends information about each one and a URL to request them from so that the client can choose.] 301 — Moved Permanently [Requested resource does not exist on the server. A Location header is sent to the client to redirect it to the new URL. Client continues to use the new URL in future requests.] 302 — Moved Temporarily [Requested resource has temporarily moved. A Location header is sent to the client to redirect it to the new URL. Client continues to use the old URL in future requests.] 303 — See Other [The requested resource can be found in a different location indicated by the Location header, and the client should use the GET method to retrieve it.] 304 — Not Modified [Used to respond to the If-Modified-Since request header. Indicates that the requested document has not been modified since the specified date, and the client should use a cached copy.] 305 — Use Proxy [The client should use a proxy, specified by the Location header, to retrieve the URL.] 307 — Temporary Redirect [The requested resource has been temporarily redirected to a different location. A Location header is sent to redirect the client to the new URL. The client continues to use the old URL in future requests.]		
		Unused status codes 306- Switch Proxy 416- Requested range not satisfiable 506- Redirection failed

Typische Datenformate im Internet:

(1) XML

```
<Kreditkarte
  Herausgeber="Xema"
  Nummer="1234-5678-9012-3456"
  Deckung="2e+6"
  Waehrung="EURO">
  <Inhaber
    Name="Mustermann"
    Vorname="Max"
    maennlich="true"
    Alter="42"
    Partner="null">
    <Hobbys>
      <Hobby>Reiten</Hobby>
      <Hobby>Golfen</Hobby>
      <Hobby>Lesen</Hobby>
    </Hobbys>
    <Kinder />
  </Inhaber>
</Kreditkarte>
```

- XML = eXtensible Markup Language
(Daten und ihre Beschreibung)
- MIME-Typ: *text/xml*, *application/xml* bzw. *application/<xml-lang>+xml*.
- Schema-Sprachen: XML Schema, DTD, Relax NG
- Data-Binding in Java: JAXB, ...
- Datentypen
 - Elemente
 - Attribute
 - Textknoten
 - Listen, Sequenzen, Auswahlen
 - 19 primitive Datentypen (string, integer, bool, ...)
 - 25 abgeleitete Datentypen (ID, IDREF, URI, ...)

Typische Datenformate im Internet:

(2) JSON

Objekt

Eigenschaft Wert

```
{  
  "Herausgeber": "Xema",  
  "Nummer": "1234-5678-9012-3456",  
  "Deckung": 2e+6,  
  "Währung": "EURO",  
  "Inhaber": {  
    "Name": "Mustermann",  
    "Vorname": "Max",  
    "männlich": true,  
    "Hobbys": [ "Reiten", "Golfen", "Lesen" ],  
    "Alter": 42,  
    "Kinder": [],  
    "Partner": null  
  }  
}
```

Array

- JSON = JavaScript Object Notation (Daten pur)
- MIME-Typ: *application/json*
- Schema-Sprachen: JSON Schema (<http://json-schema.org>)
- Data-Binding in Java: Jackson, ...
- Datentypen
 - Nullwert: `null`
 - bool'scher Wert: `true`, `false`
 - Zahl: `42`, `2e+6`
 - Zeichenkette: `"Mustermann"`
 - Array: `[1,2,3]`
 - Objekt mit Eigenschaften: `{"Name": "Mustermann"}`

Request / Response

REST ist ein Paradigma für Anwendungsservices auf Basis des HTTP-Protokolls.

- REST ist eine Paradigma für den Schnittstellenentwurf von Internetanwendungen auf Basis des HTTP-Protokolls.
- REST wurde erstmalig in der Dissertation von Roy Fielding definiert: „Architectural Styles and the Design of Network-based Software Architectures“, 2000, University of California, Irvine.
- **Grundlegende Eigenschaften:**
 - **Alles ist eine Ressource:** Eine Ressource ist eindeutig adressierbar über einen URI, hat eine oder mehrere Repräsentationen (XML, JSON, bel. MIME-Typ) und kann per Hyperlink auf andere Ressourcen verweisen. Ressourcen sind, wo immer möglich, hierarchisch navigierbar.
 - **Uniforme Schnittstellen:** Services auf Basis der HTTP-Methoden (POST = erzeugen, PUT = aktualisieren oder erzeugen, DELETE = löschen, GET = abfragen). Fehler werden über die HTTP Codes zurückgemeldet. Services haben somit eine standardisierte Semantik und eine stabile Syntax.
 - **Zustandslosigkeit:** Die Kommunikation zwischen Server und Client ist zustandslos. Ein Zustand wird im Client nur durch URIs gehalten.
 - **Konnektivität:** Basiert auf ausgereifter und allgegenwärtiger Infrastruktur: Der Web-Infrastruktur mit wirkungsvollen Caching- und Sicherheitsmechanismen, leistungsfähigen Servern und z.B. Web-Browser als Clients.



Beispiele für REST-Aufrufsyntax: Schnittstellenentwurf über Substantive.

- Produkte aus der Kategorie Spielwaren:
<http://www.service.de/produkte/spielwaren>
- Bestellungen aus dem Jahr 2008
<http://www.service.de/bestellungen/2008>
- Liste aller Regionen, in denen der Umsatz größer als 5 Mio. Euro war
<http://www.service.de/regionen/umsatz/summe?groesserAls=5M>
- Gib mir die zweite Seite aus dem Produktkatalog
<http://www.service.de/produkte/2>
- Alle Gruppen, in den der Benutzer „josef.adersberger“ Mitglied ist.
<http://www.service.de/benutzer/josef.adersberger/gruppen>

Gängige Entwurfsregeln:

- Plural, wenn auf Menge an Entitäten referenziert werden soll. Sonst singular.
- Pfad-Parameter, wenn Reihenfolge der Angabe wichtig. Sonst Query Parameter.
- Standard Query Parameter einführen (z.B. für Filter und Abfragen sowie seitenweisen Zugriff) und konsistent halten.
- Pfad-Abstieg, wenn Entitäten per Aggregation oder Komposition verbunden sind.
- Pfad-Abstieg, wenn es sich um einen gängigen Navigationsweg handelt.
- Ids als Pfad-Parameter abbilden.
- Fehler und Ausnahmen über Return Codes abbilden. Einen Standard-Code suchen, der von der Semantik her passt.

Siehe auch: <http://codeplanet.io/principles-good-restful-api-design>

WADL ist eine Syntax, mit der die Serviceschnittstelle eines REST-Webservices beschrieben werden kann.

`<application>`

`<doc/>*`

`<grammars/>?`

`<resources/>*`

`<resource-type/>*`

`<method/>*`

`<representation/>*`

`<param/>*`

`</application>`

Was kann der Dienst? Prosa.

Welche Definitionen für Datenformate werden verwendet? z.B. XSD, JSON Schema.

**Welche Ressourcen werden angeboten?
Ihr Verhalten: URL-Pattern, Parameter.**

**Welche Typen an Ressourcen mit
gemeinsamen Verhalten gibt es?**

**Was ist das Default-Verhalten
für eine HTTP-Methode?**

**Welche wiederverwendbaren Deklarationen
von Repräsentationsformaten gibt es?**

**Welche wiederverwendbaren Parameter-
Deklarationen gibt es?**

siehe: <http://wadl.java.net/wadl20090202.pdf>

REST-Webservices mit JAX-RS.



<http://www.service.de/hello/Josef?salutation=Servus>



@GET, @POST, @PUT, @DELETE

```
@Path("/hello/{name}")
public class HelloWorldResource {

    @GET
    @Produces("application/json")
    public ResponseMessage getMessage(
        @DefaultValue("Hallo") @QueryParam("salutation") String salutation,
        @PathParam("name") String name) throws IOException {
        ResponseMessage response = new ResponseMessage(new Date().toString(), salutation + " " + name);
        return response;
    }
}
```

← Analog @Consumes für 1. Parameter

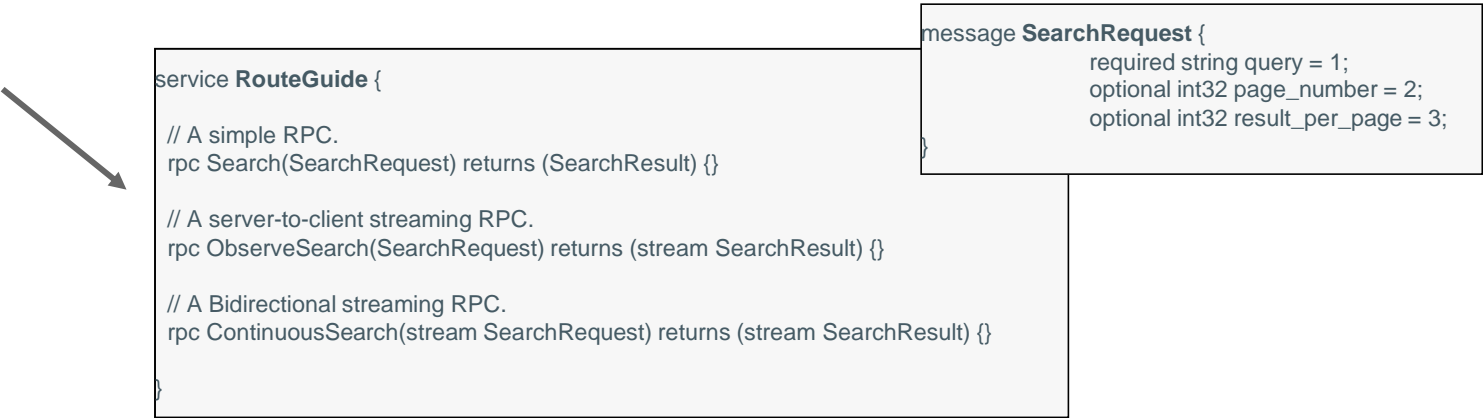
← Analog @FormParam bei POST Requests

Die effizienten Alternativen: Binärprotokolle

- Binärprotokolle sind eine sinnvolle Alternative zu REST/SOAP, wenn eine effiziente und programmiersprachennahe Kommunikation im Backend erfolgen soll.
- Encoding der Payload als komprimiertes Binärformat
- Separate Schnittstellenbeschreibungen (IDLs) aus denen dann Stubs und Skeletons in mehreren Programmiersprachen generiert werden können

■ Kandidaten

- GRPC (HTTP2!)
- Apache Avro
- Apache Thrift
- Hessian
- BSON



```
service RouteGuide {  
    // A simple RPC.  
    rpc Search(SearchRequest) returns (SearchResult) {}  
  
    // A server-to-client streaming RPC.  
    rpc ObserveSearch(SearchRequest) returns (stream SearchResult) {}  
  
    // A Bidirectional streaming RPC.  
    rpc ContinuousSearch(stream SearchRequest) returns (stream SearchResult) {}  
}
```

```
message SearchRequest {  
    required string query = 1;  
    optional int32 page_number = 2;  
    optional int32 result_per_page = 3;  
}
```

Push

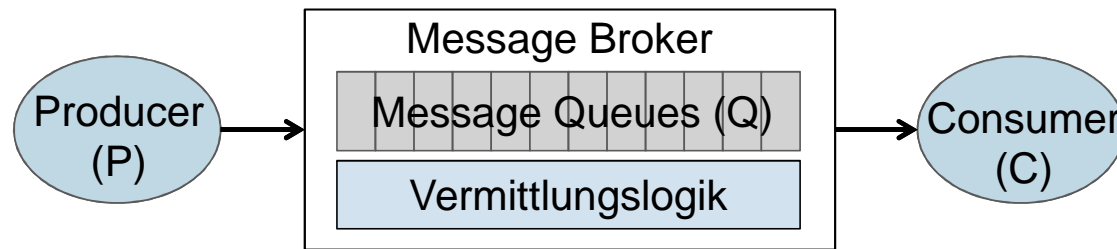
WebSockets



- Über das HTTP-Protokoll ist es nicht direkt möglich, dass ein Server aktiv eine Verbindung mit einem Client aufbaut und mit diesem kommuniziert. Es gibt eine Reihe an Workarounds, die unter dem Begriff COMET zusammengefasst sind, die dies durch technische Kniffe umgehen.
- WebSockets ist ein standardisiertes Protokoll auf Basis TCP, das über einen HTTP-Request initiiert werden kann und dann über den HTTP-Port (i.d.R. 80) per Multiplexing läuft. Das Protokoll hat aber in der Kommunikation einen geringeren Overhead im Vergleich zu HTTP.
- Ein WebSocket ist ein bidirektionaler und dauerhafter Kommunikationskanal zwischen Client (typischerweise einem Browser) und einem Server im Web. Der Kanal kann über TLS/SSL verschlüsselt sein.
- Über einen WebSocket können wiederum beliebige höhere Protokolle getunnelt werden (z.B. AMQP).

Messaging

Messaging ist zuverlässiger, asynchroner Nachrichtenaustausch.



■ Entkopplung von Producer und Consumer.

Die Serviceschnittstelle ist lediglich das Format der Nachricht. Message Broker machen zum Format keinen Einschränkungen. Sende-Zeitpunkt und Empfangs-Zeitpunkt können beliebig lange auseinander liegen.

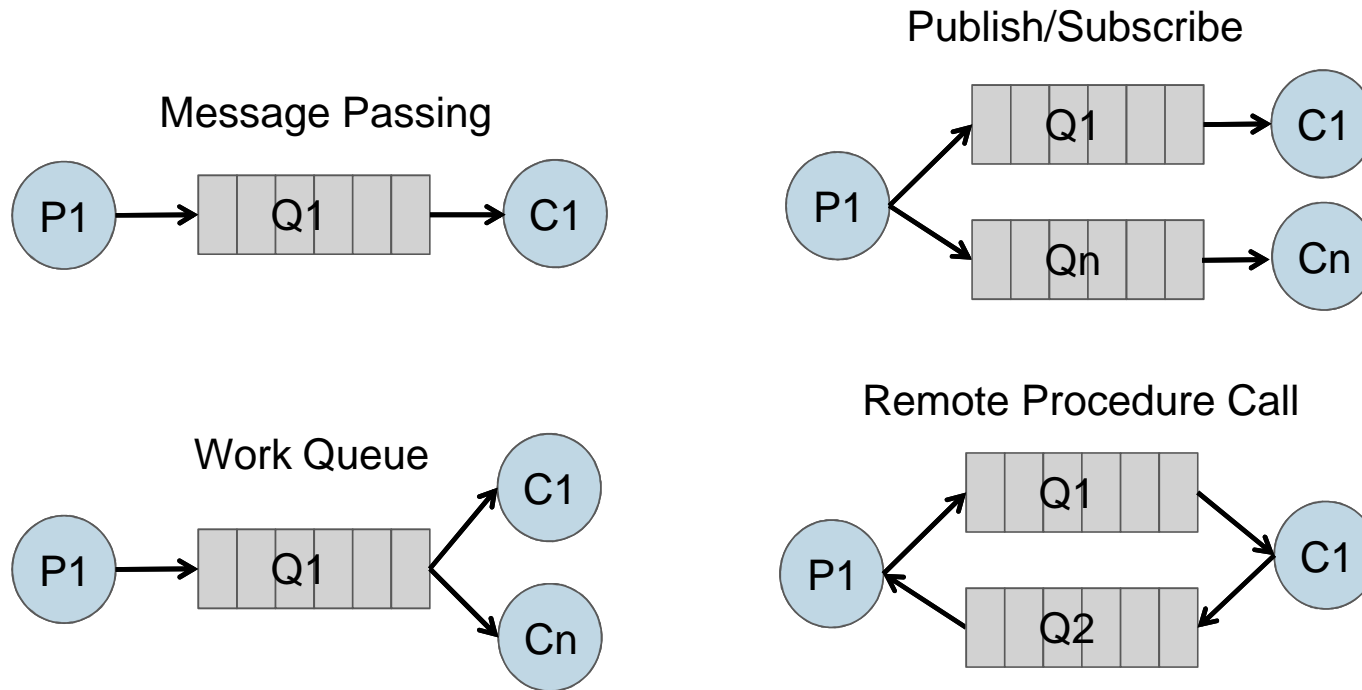
■ Skalierbarkeit. Die Vermittlungslogik entscheidet zentral ...

- ... an wie viele Consumer die Nachricht ausgeliefert wird (horizontale Skalierbarkeit),
- an welchen Consumer die Nachricht ausgeliefert wird (Lastverteilung),
- wann eine Nachricht ausgeliefert wird (Pufferung von Lastspitzen),

auf Basis von konfigurierten Anforderungen an die Vermittlung:

- Maximale Zustelldauer bzw. Lebenszeit der Nachricht
- Geforderte Zustellgarantie (mindestens 1 Mal, exakt 1 Mal, an alle) und Transaktionalität
- Priorität der Nachricht
- Notwendige Einhaltung der Zustellreihenfolge

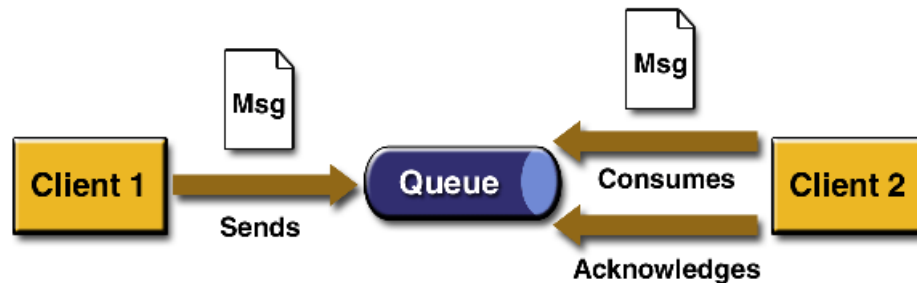
Messaging ist eine flexible Kommunikationsart, mit der sich vielfältige Kommunikationsmuster umsetzen lassen.



JMS

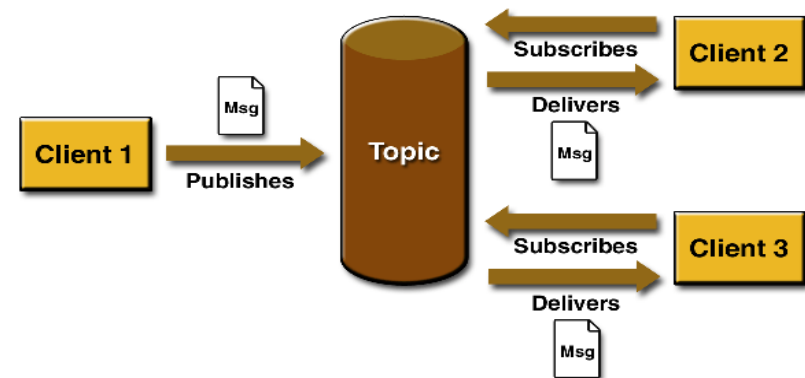
- JMS = Java Messaging Service. Standardisierte API im Rahmen der Java-Enterprise-Edition-Spezifikation. Standardisiert nicht das Messaging-Protokoll.
- 2002-2013: Version 1.1. Sehr stabil und weit verbreitet in der Java-Welt.
- Seit Mai 2013: Version 2.0 als Teil der JEE 7 Spezifikation
- Unterstützte Kommunikationsmuster:

Message Passing:



- Ein Consumer pro Message
- Der Erhalt einer Nachricht wird bestätigt

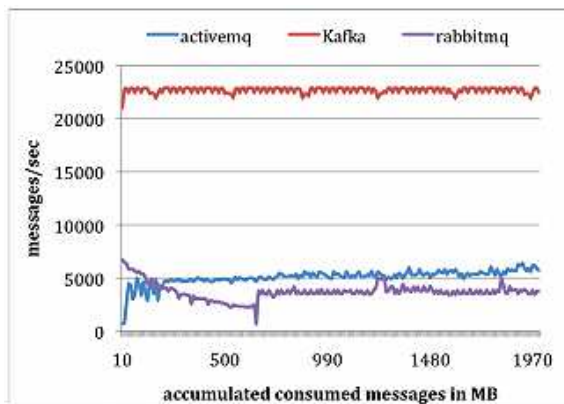
Publish / Subscribe:



- Mehrere Consumer pro Message

Kafka

■ Kafka ist schnell:



■ Entwickelt bei LinkedIn

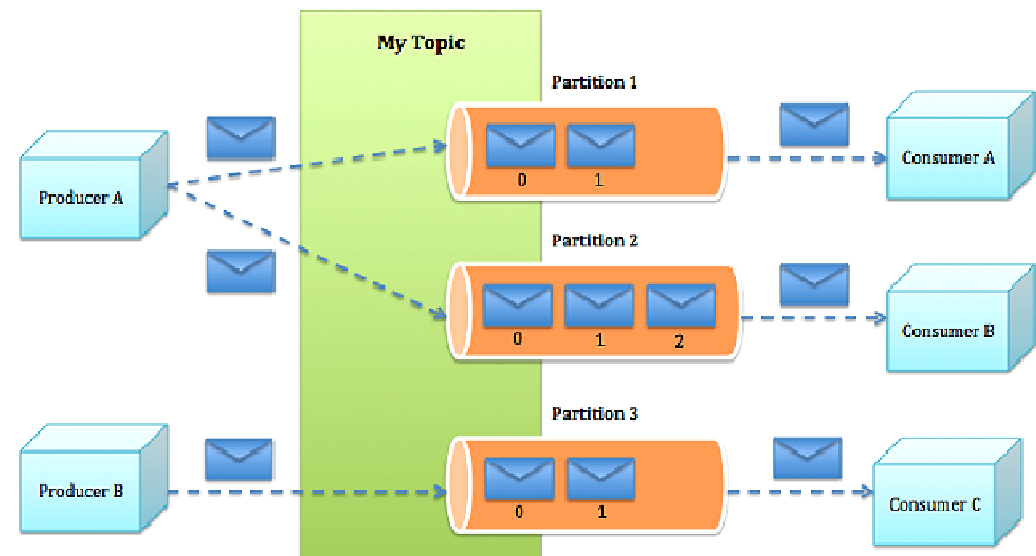
■ Die API ist proprietär

■ Die gute Idee: Message Queue als verteilter Log umgesetzt

■ Der de-facto Standard in der Cloud

<http://www.infoq.com/articles/apache-kafka>

Kafka ist hochgradig verteilbar:



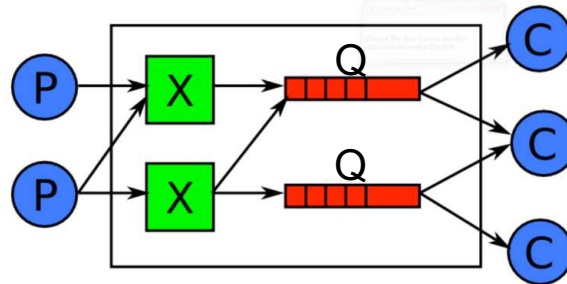
AMQP: Ein Standard-Protokoll für Messaging-Systeme.

- **Problem:** Message Broker sind intern proprietär aufgebaut (Beispiel: IBM MQSeries mit 80% Marktanteil im kommerziellen Bereich). Sie sind nicht zueinander operabel, wie man es z.B. von SNMP-Servern her kennt. Das ist besonders beim Messaging über Firmengrenzen und Technologie-Stacks hinweg ein Problem.



- **Lösung AMQP:** Standardisierung eines interoperablen Protokolls für Messaging-Broker. AMQP steht seit Ende 2011 in der Version 1.0 zur Verfügung.
 - Im Standardisierungsgremium sind u.A. Cisco, Microsoft, Red Hat, Deutsche Börse Systems, IONA, Novell, Credit Suisse, JPMorganChase.
 - Standardisiert ein Netzwerk-Protokoll für die Kommunikation zwischen den Clients und den Message Brokern.
 - Standardisiert ein Modell der verfügbaren APIs und Bausteine für die Vermittlung und Speicherung von Nachrichten.

Das AMQP-Modell definiert Standard-Bausteine eines Messaging-Systems und ihr Zusammenwirken.



- Nachrichten werden von einem **Producer (P)** erzeugt und bei einem **Exchange (X)** angeliefert.
 - Ein Exchange repräsentiert die Vermittlungslogik von Nachrichten. Eine Nachricht kann diverse Attribute besitzen, die einer Exchange als Metadaten dienen können. AMQP definiert verschiedene Exchange-Typen. Default-Exchange ist ein Routing auf Basis des Routing-Schlüssels einer Nachricht.
- **Consumer (C)** erzeugen **Queues (Q)** und binden diese an Exchanges.
 - Ein Consumer kann sich neue Nachrichten bei Bedarf an seiner Queue abholen, oder sich neue Nachrichten direkt anliefern lassen.
 - Eine Queue ist ein FIFO-Puffer für Nachrichten von potenziell unendlicher Größe. FIFO ist nur dann garantiert, wenn die Queue nur von einem Consumer genutzt wird. Eine Queue kann Nachrichten bei Bedarf persistieren. Optional kann sie auch alle Nachrichten persistieren. Eine Nachricht wird aus einer Queue heraus immer nur genau einem Consumer erfolgreich zugestellt.

Die AMQP Exchange-Typen

Direct Exchange (default):

Message mit Routing-Schlüssel an Queue gebunden an diesen Schlüssel.

- Uses string as routing key
- Queue binds to exchange with key K
- Publisher sends message with key R
- Message is passed to this queue if K=R
- Direct exchange named "amq.direct" is always pre-created in each virtual host

Topic Exchange:

Message mit Routing-Schlüssel in Queue, deren Wildcard-Binding passt.

- Uses pattern as routing key ("a.b.c.d")
- Queues can use wildcard characters in binding
- * matches a single word, # - zero or more words
- "amq.topic" is pre-created in each vhost
- *.stock.# matches usd.stock and eur.stock.db but not stock.nasdaq

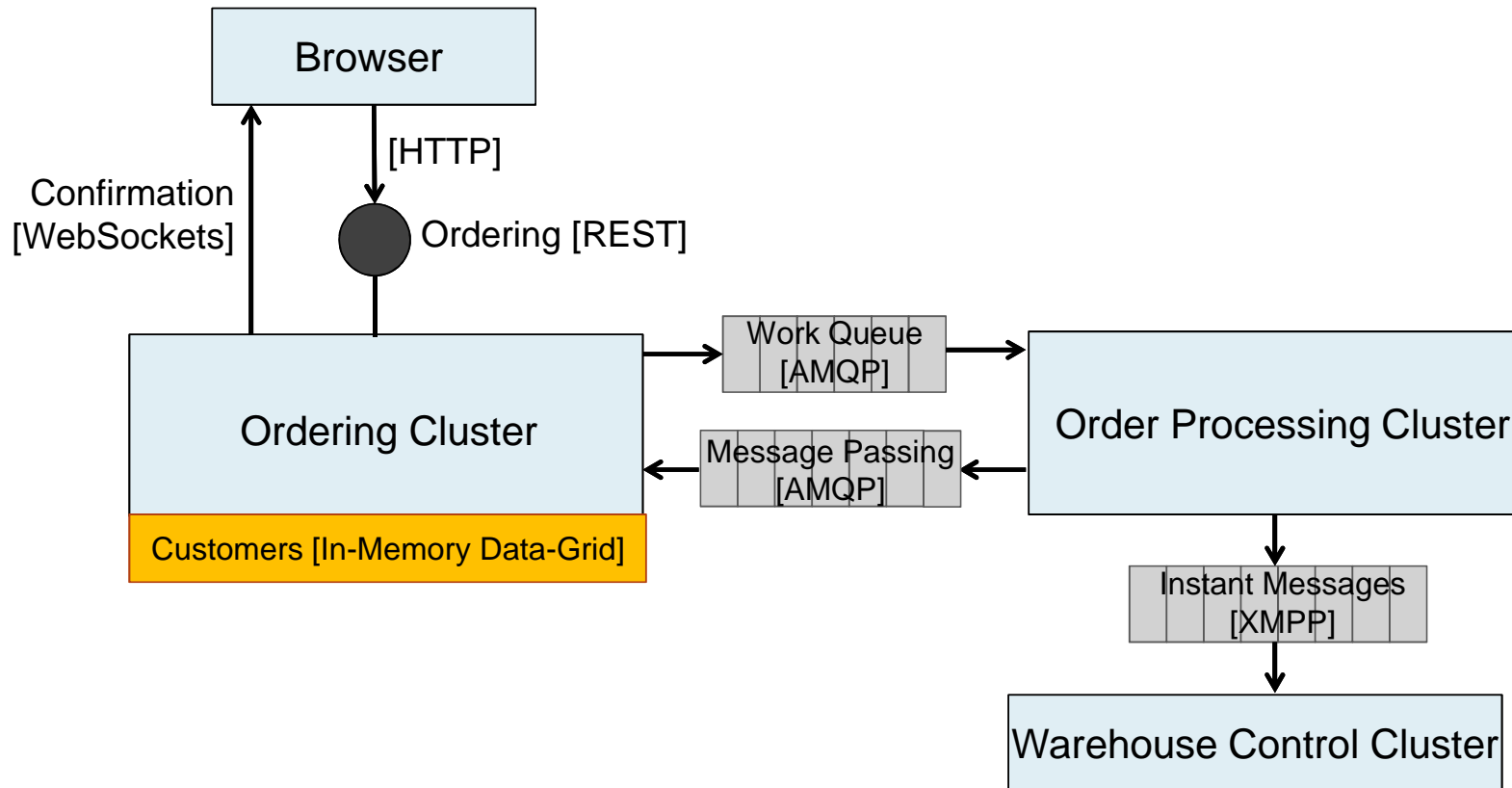
Fanout Exchange:

Alle Nachrichten wandern in alle gebundenen Queues.

- No routing key
- What goes in must go out
- Can be used for load balancing

Zusammenfassung

Putting it all together...



Literatur

Literatur

■ Bücher:

- Patterns of Enterprise Application Architecture, Martin Fowler, 2002
- Computer Networks, Andrew Tanenbaum, 2010
- Inter-Process Communication, Hephaestus Books, 2011

■ Internet:

- REST und SOAP im Vergleich (<http://www.oio.de/public/xml/rest-webservices.htm>)
- Enterprise Integration Patterns (<http://camel.apache.org/enterprise-integration-patterns.html>)
- Dissertation von Roy Fielding zu REST
(http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm)
- RESTful Webservices (<http://www.ibm.com/developerworks/webservices/library/ws-restful>)