

31.8.–3.9.2015
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Sie haben Post!

Mobiles polyglottes Messaging

Frank Pientka

MATERNA GmbH

Sie haben Post



Mobiles polyglottes Messaging
Frank Pientka, Dortmund

Wer ist Frank Pientka?

Dipl.-Informatiker (TH Karlsruhe)

Principal Software Architect in Dortmund

iSAQB-Gründungsmitglied

heise.de/developer/Federlesen-Kolumne

Über 20 Jahre IT-Erfahrung
Veröffentlichungen und Vorträge



Frank Pientka
frank.pientka@materna.de
+49 (231) 5599 8854
+49 (1570) 1128854
www.materna.de

Agenda

Kommunikation
verändert sich

Mobiles
Messaging –
Möglichkeiten &
Heraus-
forderungen

Messaging-
Protokolle –
MQTT &
STOMP

Messaging-
Produkte für
MQTT &
STOMP

Beispiel -
JavaScript,
Java, CLI

Fazit –
wird möglich

Kommunikation verändert sich



Web 1.0: 1 Sender, viele Empfänger

Web 2.0: viele Sender, viele Empfänger

**Mobiles Web, IoT, M2M:
immer & überall & jeder & jedes**

Mobile ist die Zukunft (2014)

300 Mio



Verkaufte PCs

2,2 Mrd



Verkaufte
Smartphones, Tablets

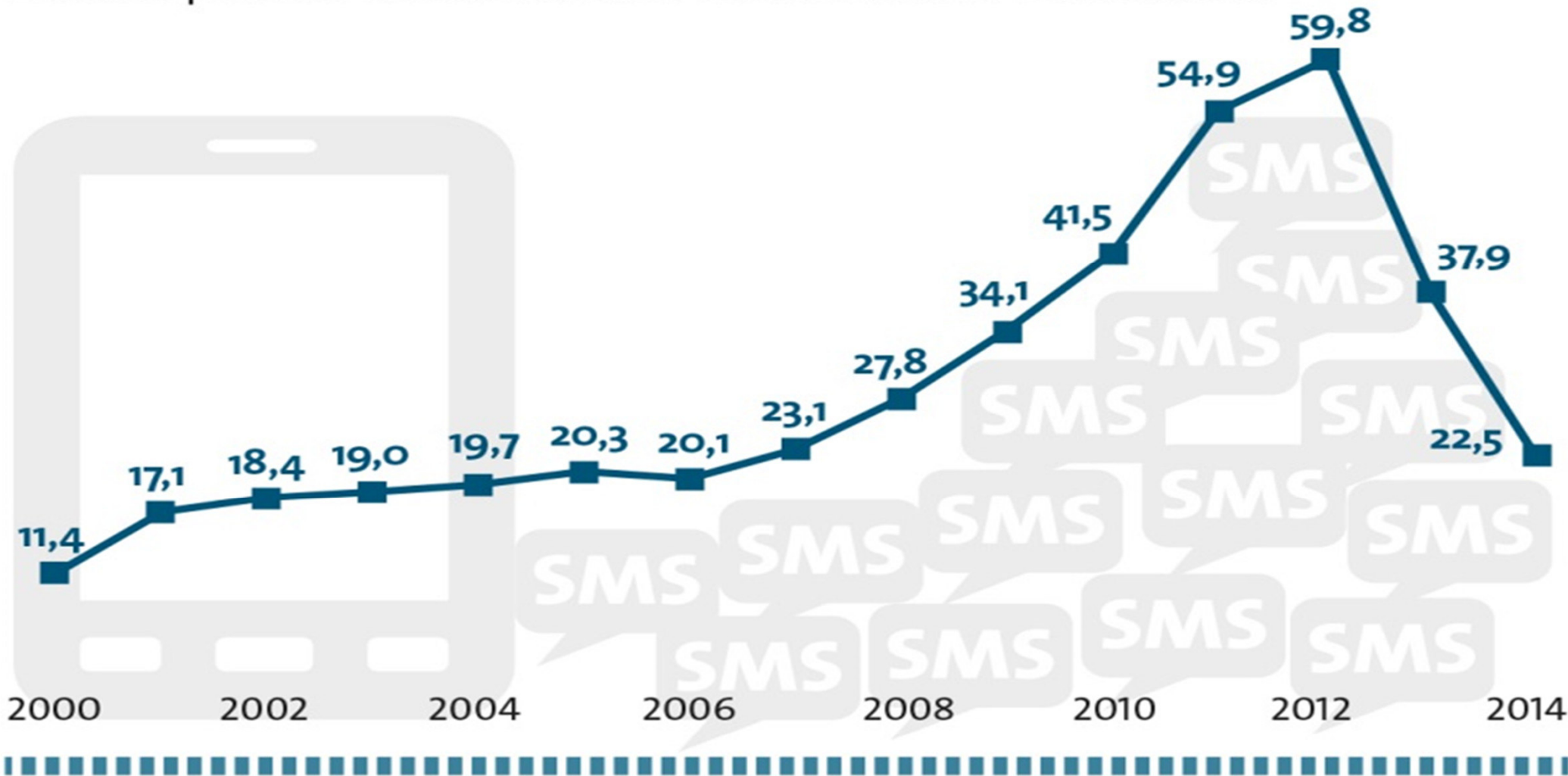
7,2 Mrd



Menschen

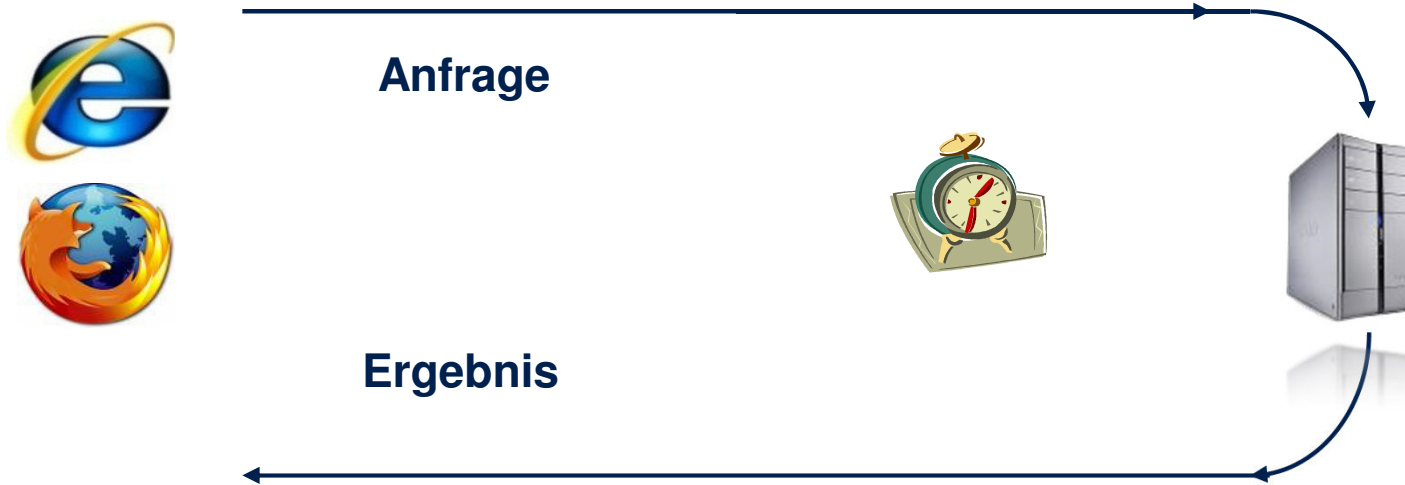
Die Talfahrt geht weiter

Zahl der pro Jahr versandten SMS in Deutschland in Milliarden

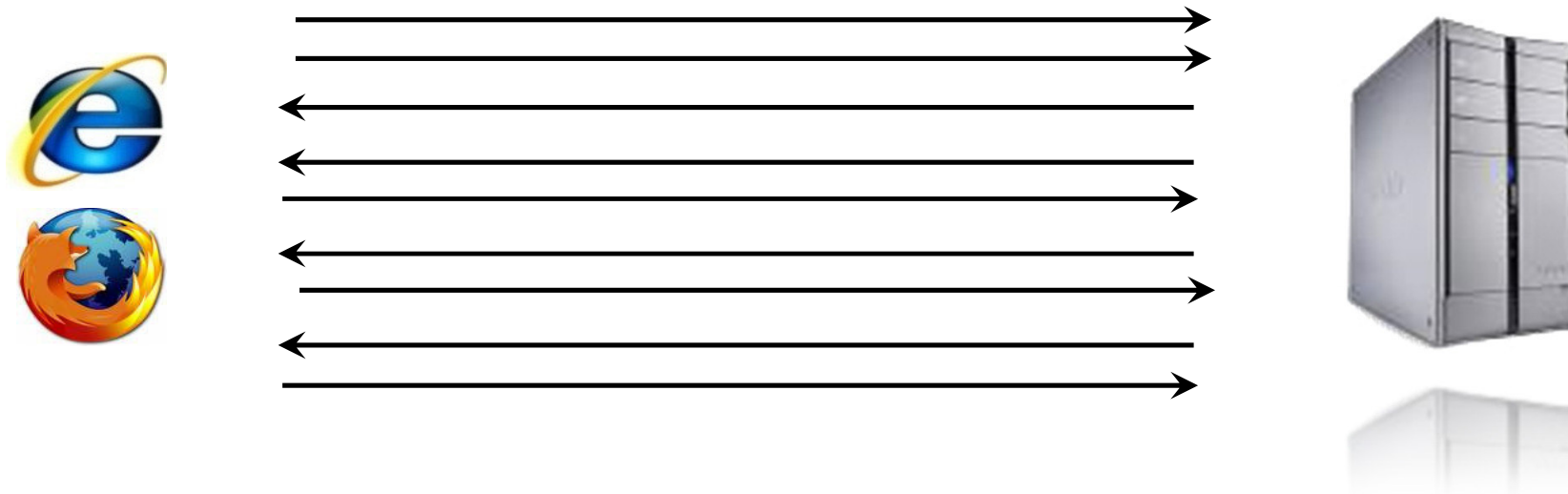


Das synchrone Web ..

...war gestern



Heute: Ereignisorientierte Verarbeitung



Mobiles Messaging Quo vadis?



Agenda

Kommunikation
verändert sich

Mobiles
Messaging –
Möglichkeiten &
Heraus-
forderungen

Messaging-
Protokolle –
MQTT &
STOMP

Messaging-
Produkte für
MQTT &
STOMP

Beispiel
JavaScript,
Java, CLI

Fazit –
wird möglich

Mobile App-Arten und Technologien

Web

STOMP
MQTT

CSS3
JS
HTML5

Hybride

XMPP
STOMP
MQTT

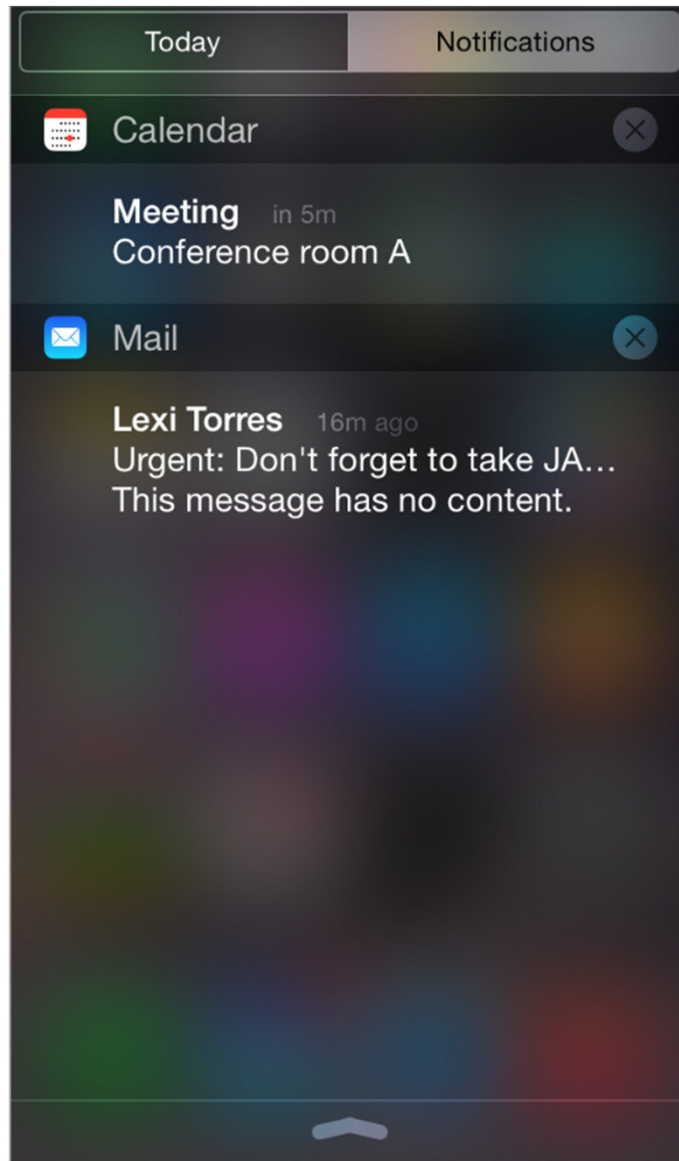
Xamarin
Cordova

Native

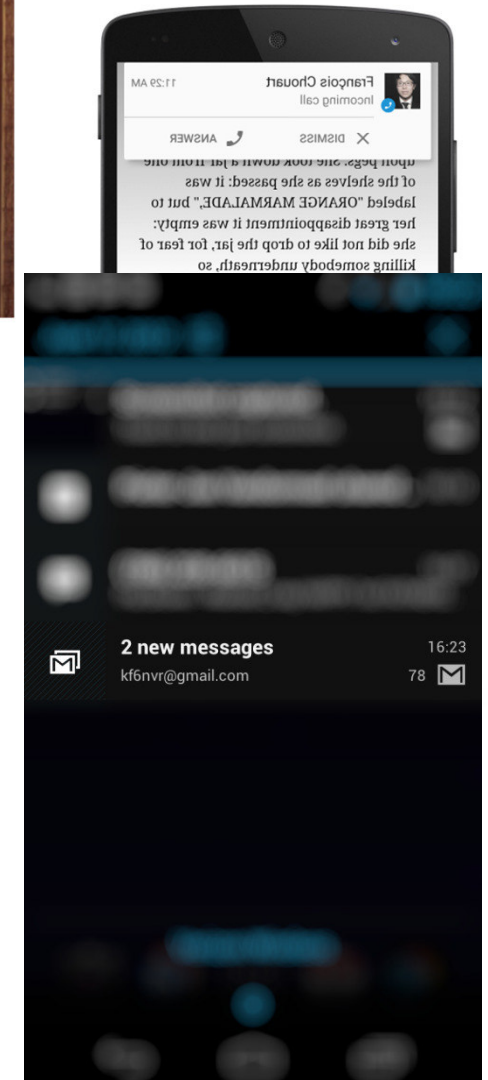
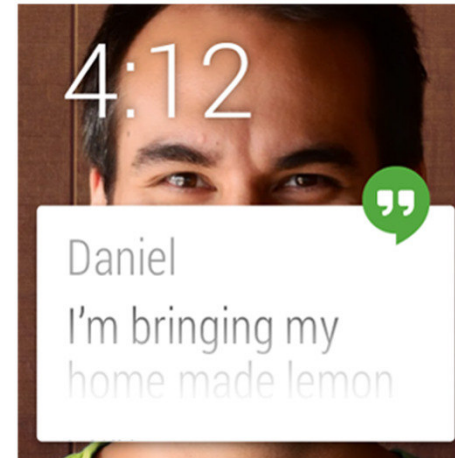
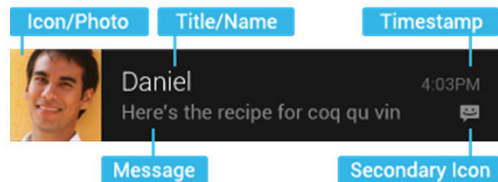
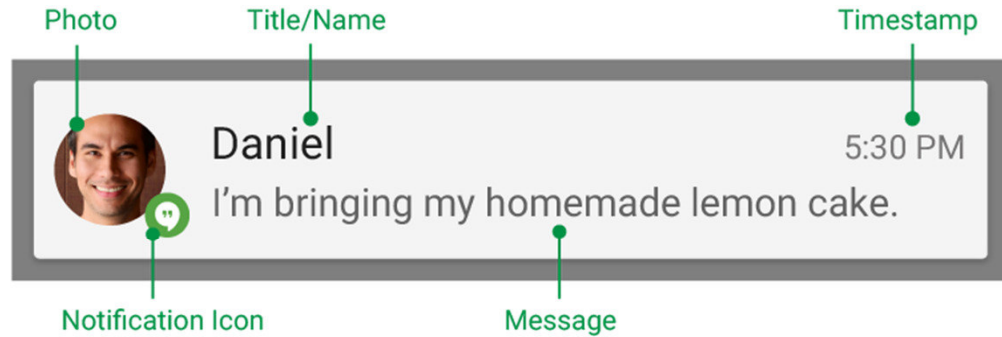
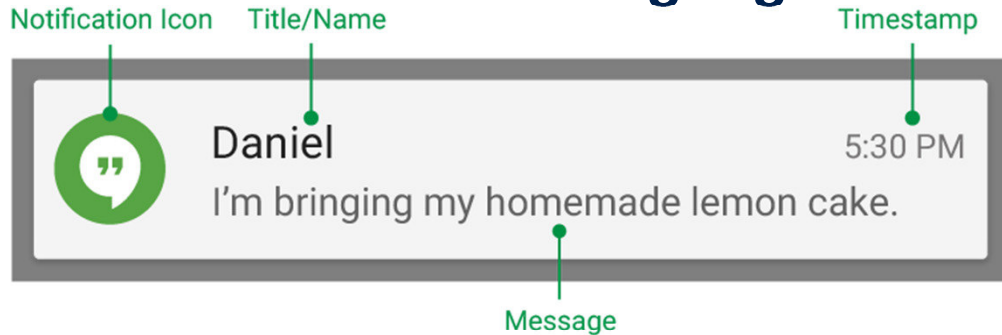
XMPP
STOMP
MQTT

Objective-C
Java

Benutzer Benachrichtigung - iPhone



Benutzer Benachrichtigung - Android



Herausforderungen mobile Kommunikation



Herausforderungen



Registrierung

Heterogenität

QoS

Akku

Latenz

Innovation

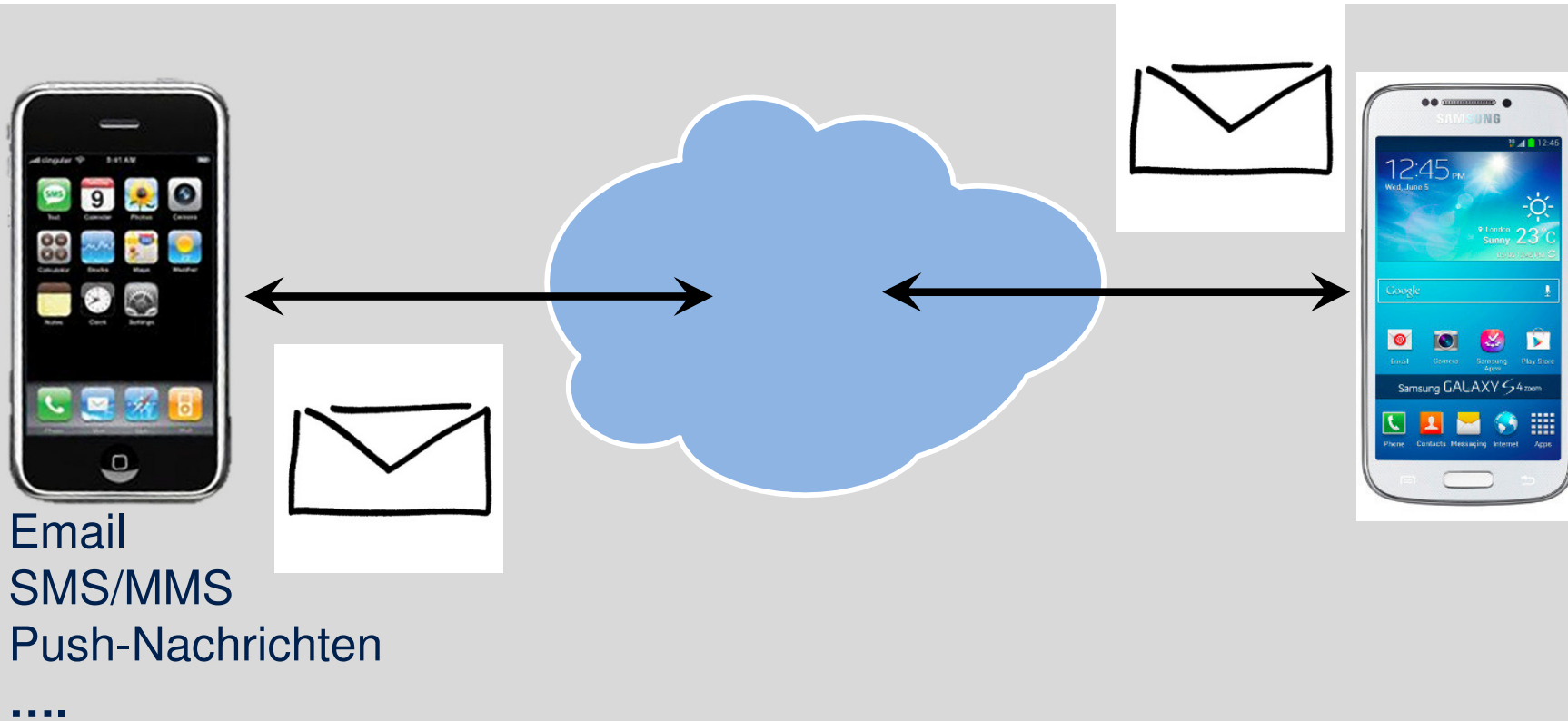
Notification

Robustheit

Sicherheit

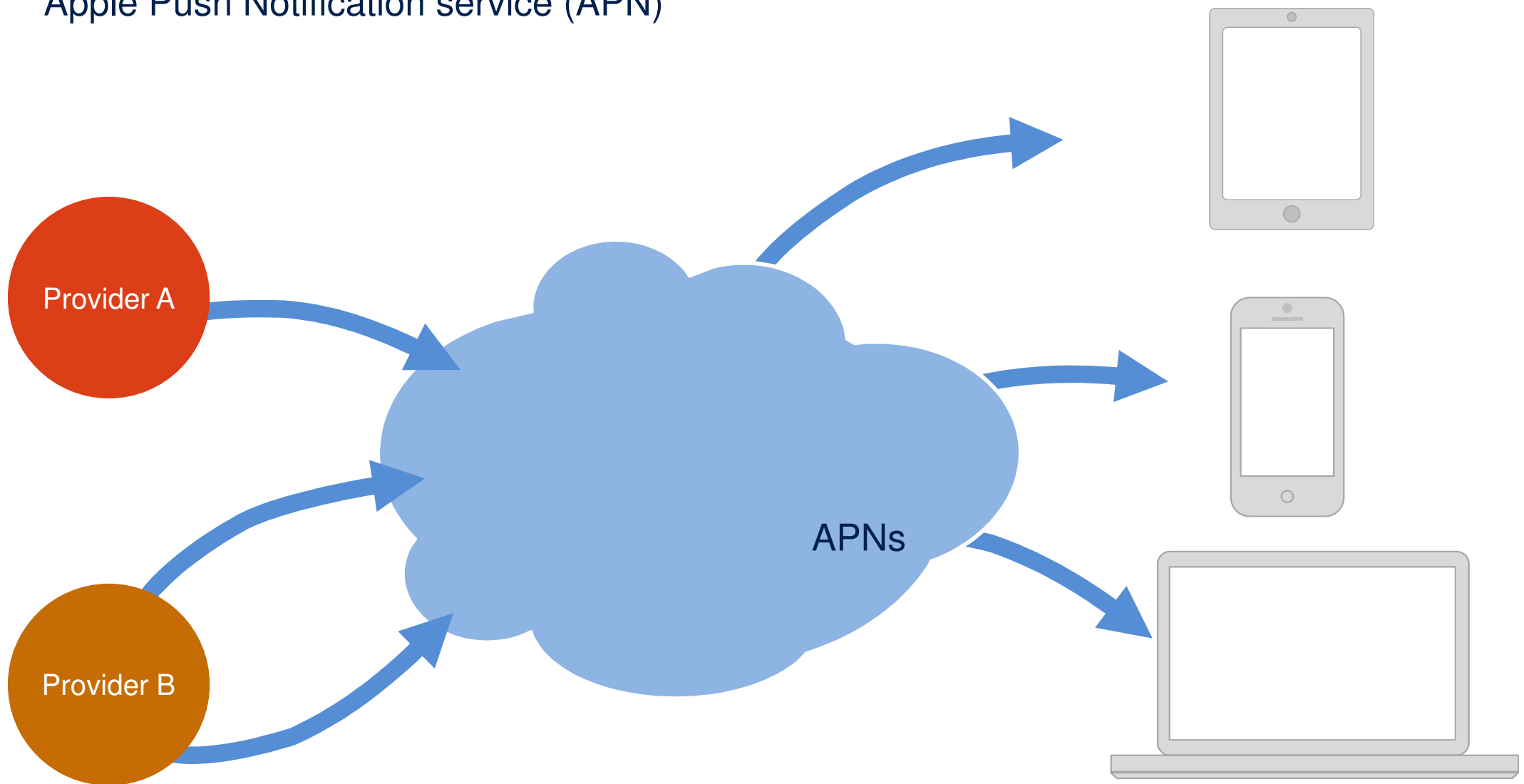
Mobiles Messaging

Mobile Nachrichten Überblick

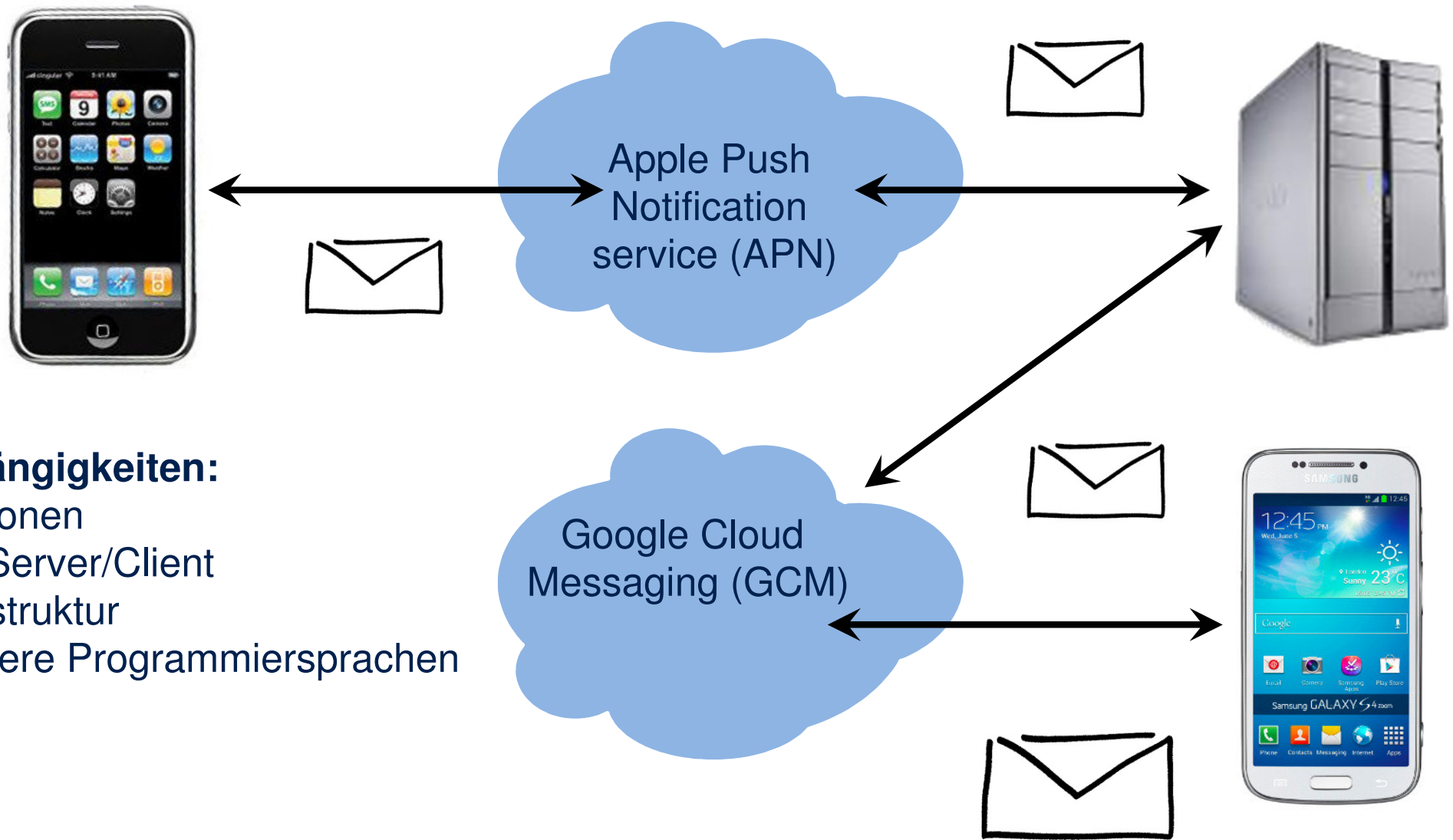


Mehrere Anbieter, mehrere Nutzer

Apple Push Notification service (APN)



Mobile Nachrichten mit iOS und Android



Abhängigkeiten:

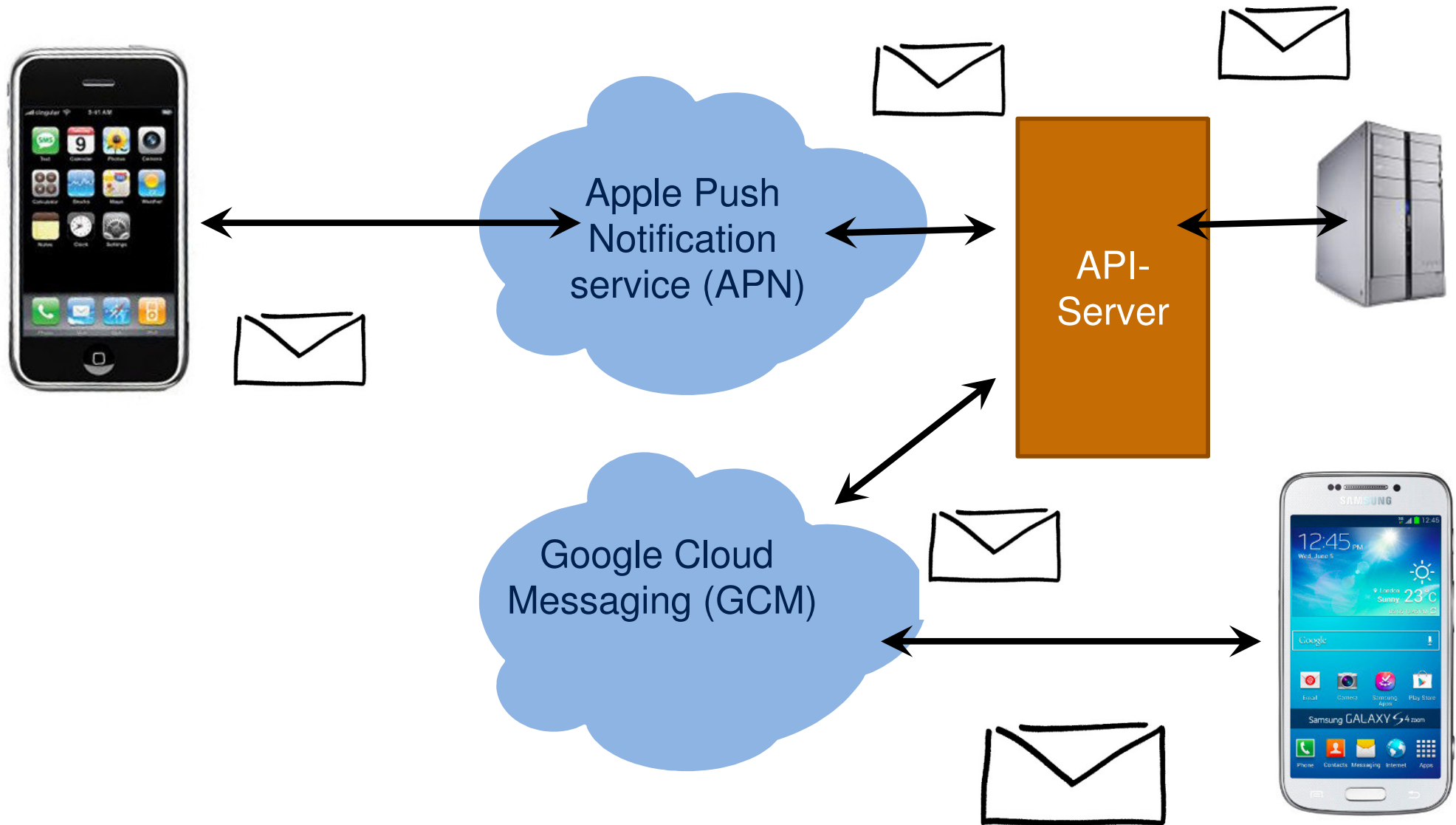
Versionen

API-Server/Client

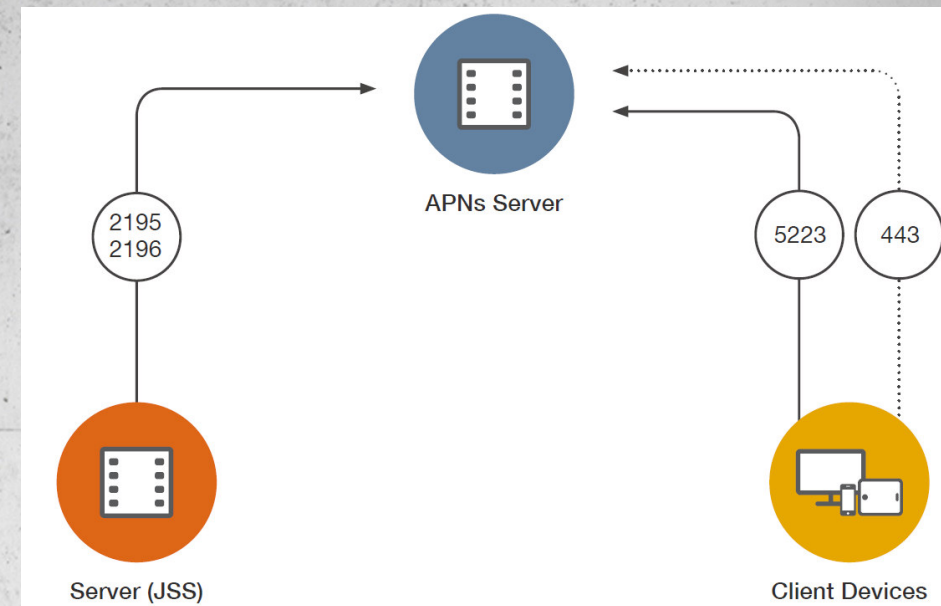
Infrastruktur

Mehrere Programmiersprachen

Mobile Nachrichten – vereinheitlichte Server-API



Apple Push Notification service (APN) Ports



TCP port 5223 (used by devices to communicate)

TCP port 2195 (used by devices to send notifications)

TCP port 2196 (used by the APNs feedback service)

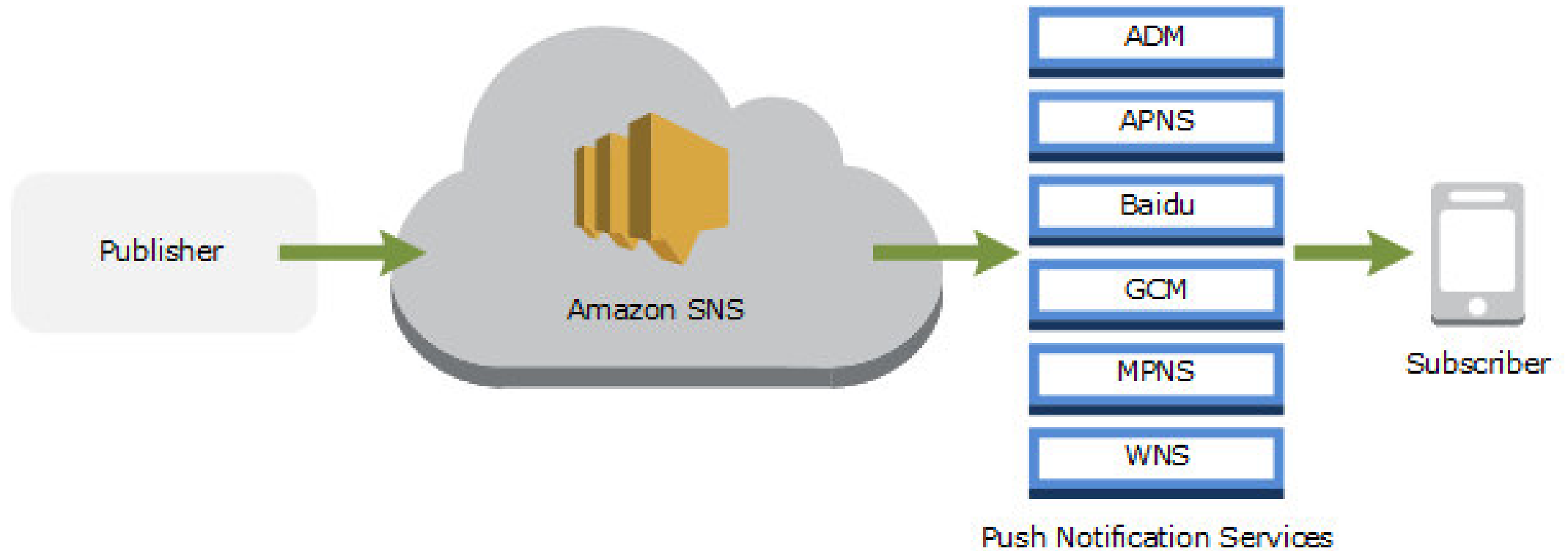
TCP Port 443 (Wi-fi fallback, when unable to communicate on port 5223)

Google Cloud Messaging Kommunikationsarten

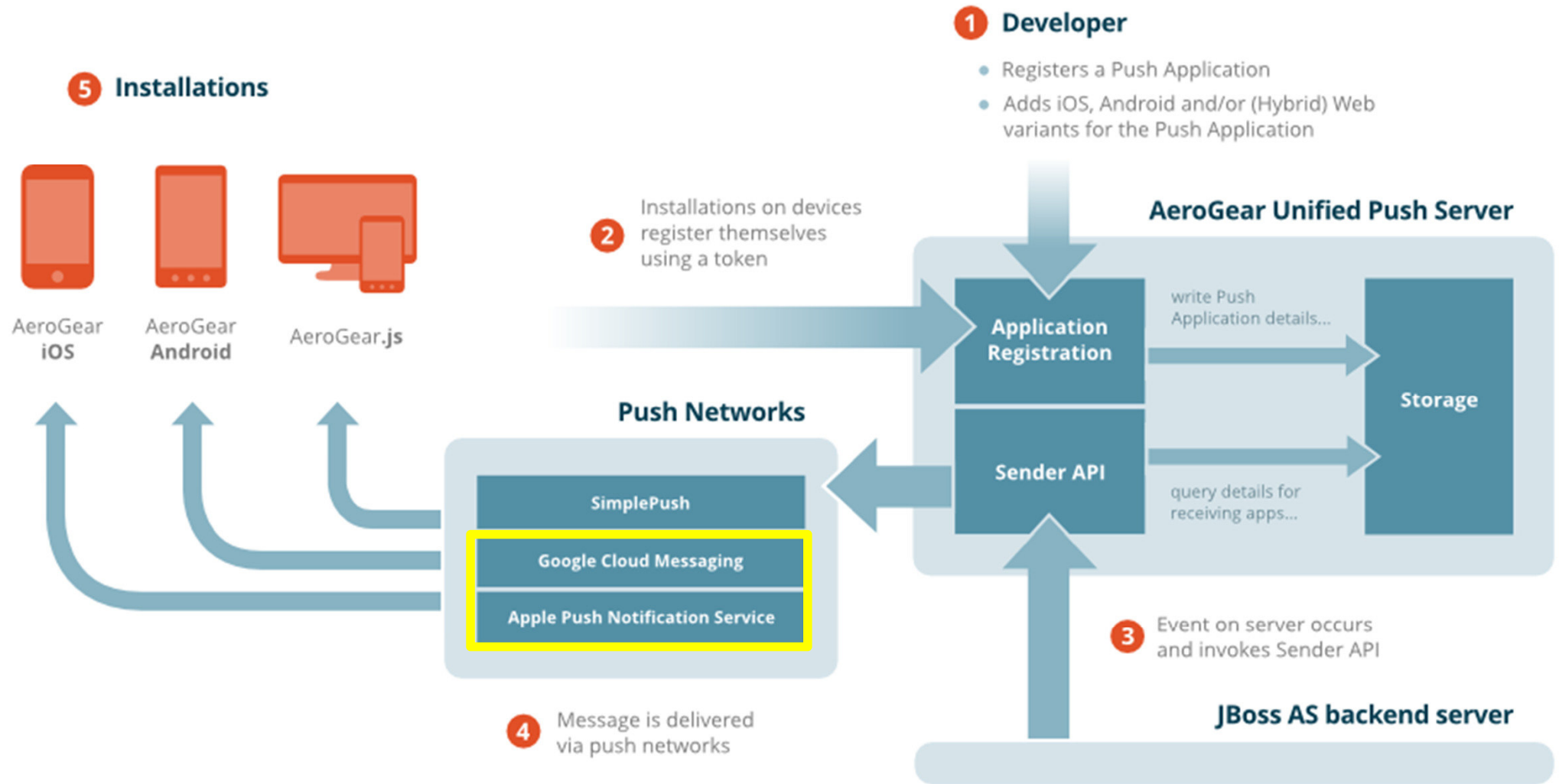
Cloud Connection Server (CCS)

- Upstream/Downstream messages
 - HTTP: Downstream only, cloud-to-device
 - XMPP: Upstream and downstream (device-to-cloud, cloud-to-device)
- Messaging (synchronous, asynchronous)
 - HTTP POST synchronous requests and wait for a response
 - XMPP: Asynchronous send/receive messages to/from devices at full over **persistent XMPP** connections. sends **acknowledgment** or failure notifications
- JSON
 - HTTP: JSON messages sent as HTTP POST
 - XMPP: JSON messages encapsulated in XMPP messages.
- Plain Text
 - HTTP: Plain Text messages sent as HTTP POST
- Multicast downstream send to multiple registration IDs
 - HTTP: Supported in JSON message format
- **Deletion** of already delivered messaging possible

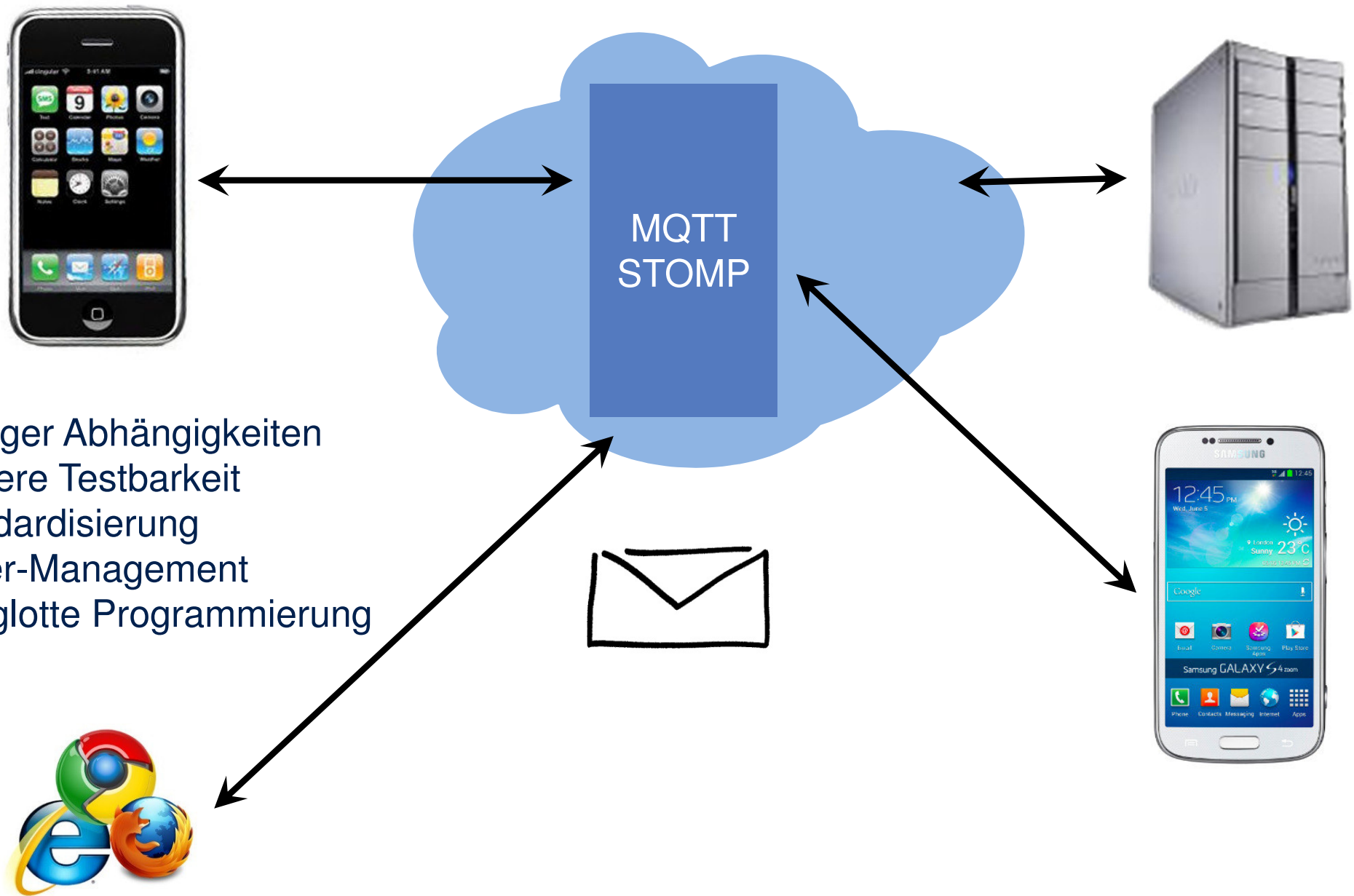
Lösung - Amazon SNS



Lösung - AeroGear Unified Push Server



Lösung - Mobile Nachrichten – vereinheitlichte Server-/Client-API/Protokoll



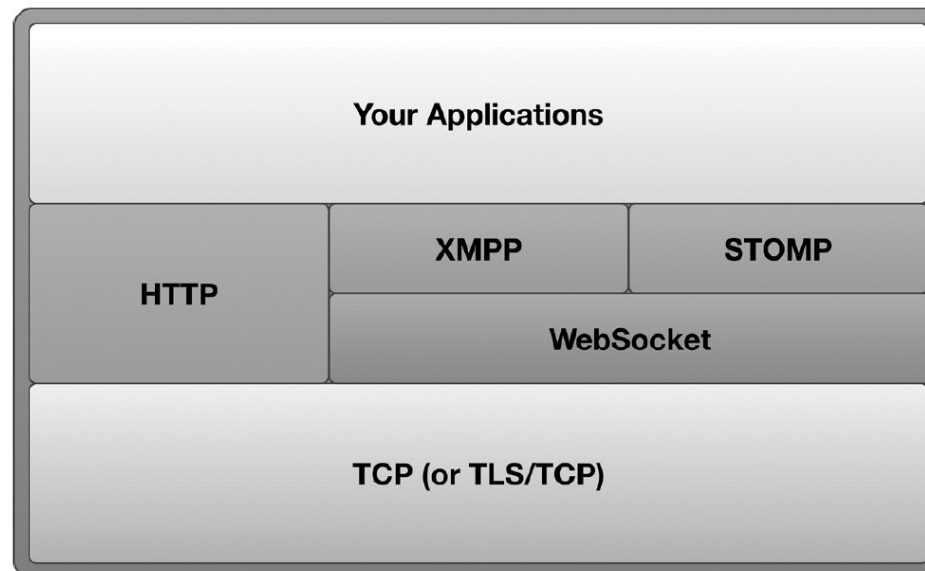
- +Weniger Abhängigkeiten
- +bessere Testbarkeit
- +Standardisierung
- Server-Management
- Polyglotte Programmierung

Agenda



Nachrichtenprotokolle

- **JMS** (Java Message Service) → JCP
- **AMQP** (Advanced Message Queuing Protocol) → OASIS, ISO/IEC 19464
- **OpenWire** → Apache License 2.0
- **STOMP** (Streaming Text Oriented Messaging Protocol) → (CC BY 3.0)
- **XMPP** (Jabber) → RFC/IETF
- **MQTT** (Message Queue Telemetry Transport) → OASIS



Nachrichtenprotokolle Einteilung

Mehrsprachig (Client/Server)

STOMP (TEXT)
XMPP (XML/JSON)#

AMQP#
MQTT

Text

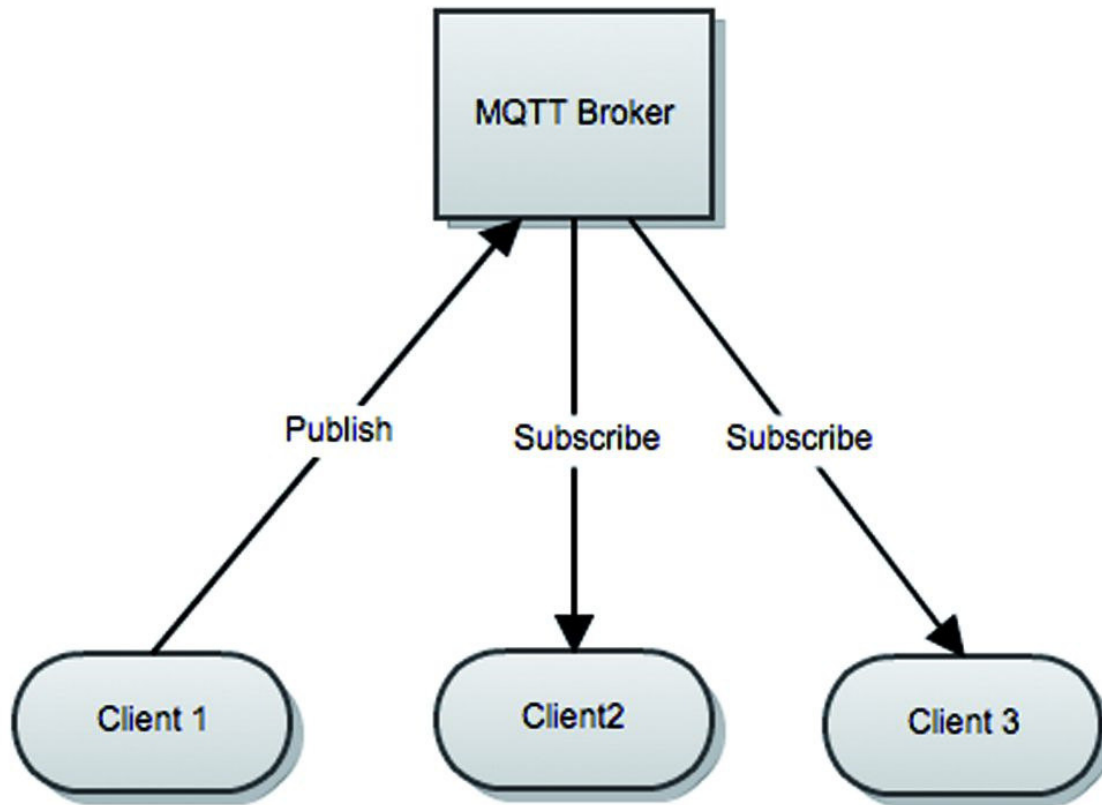
binär

HTTP/S-Unterstützung

JMS#
OpenWire

singulär

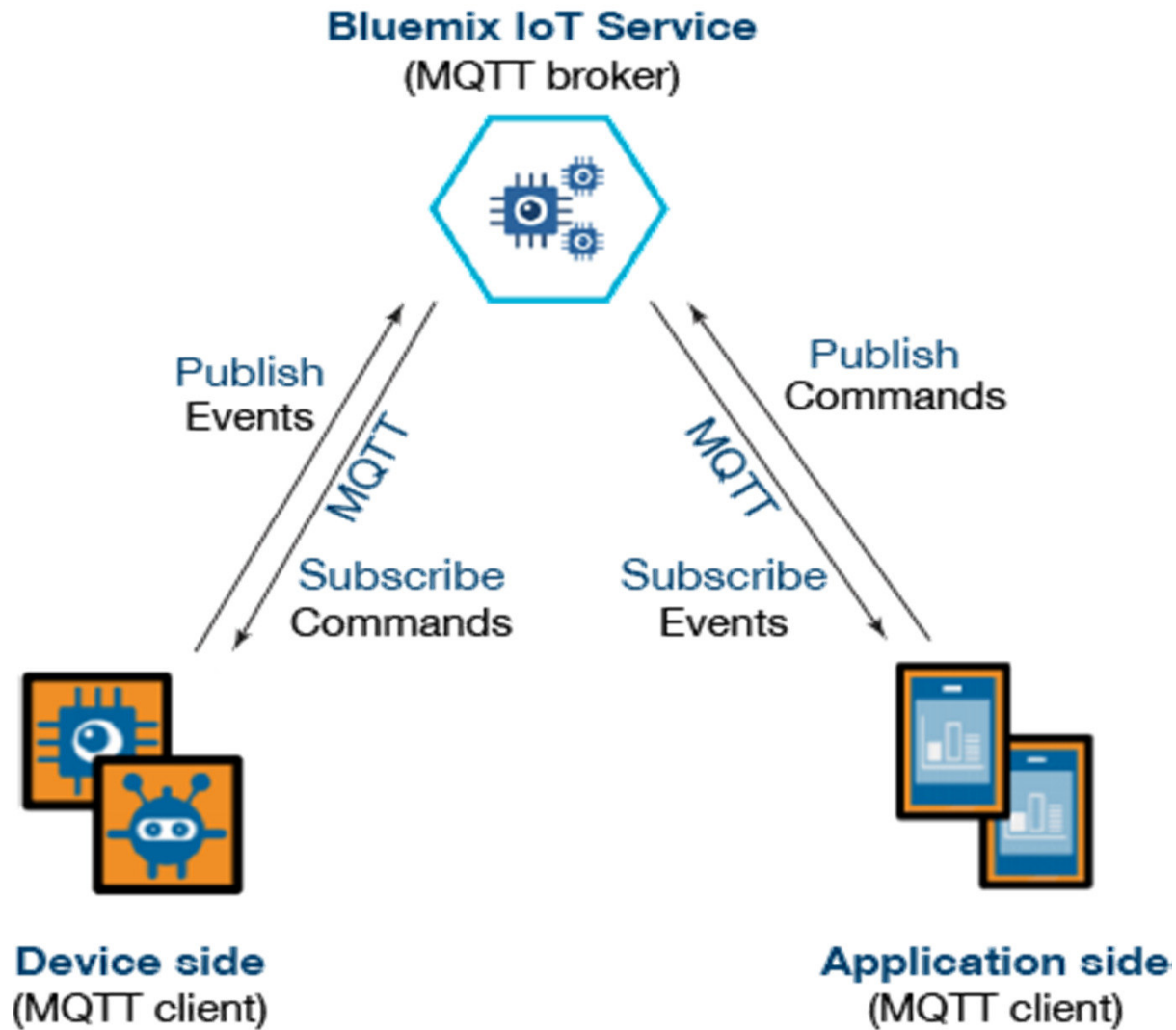
Message-Queue-Telemetry-Transport-(MQTT-)Protokoll



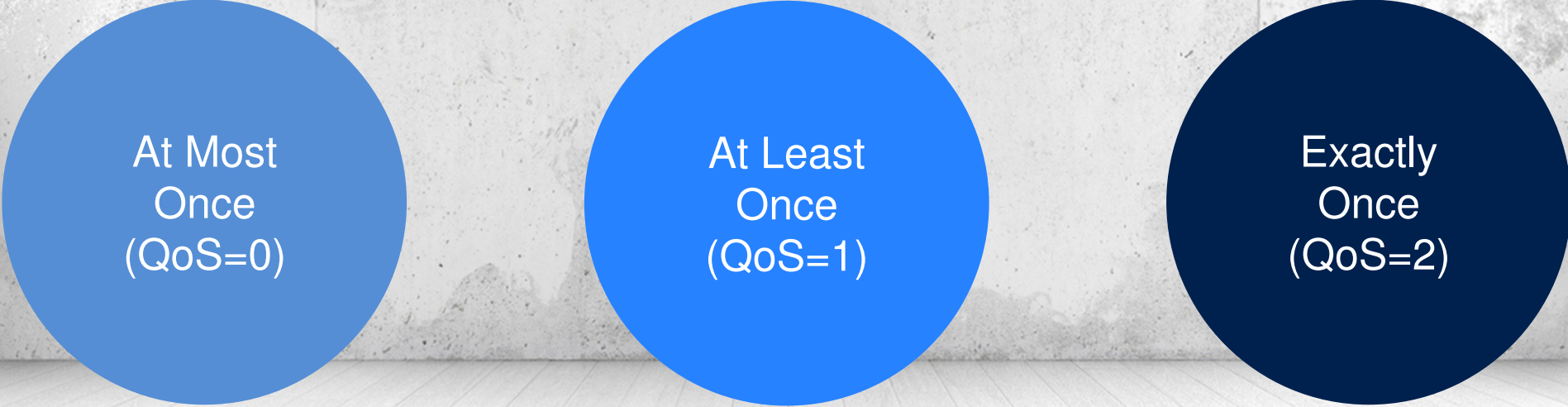
- **1999** von Andy Stanford-Clark (IBM) und Arlen Nipper (Eurotech)
- **2014** MQTT Protocol Specification Version 3.1.1



MQTT-Überblick Publish-Subscribe-Broker



MQTT Quality of Service (QoS)



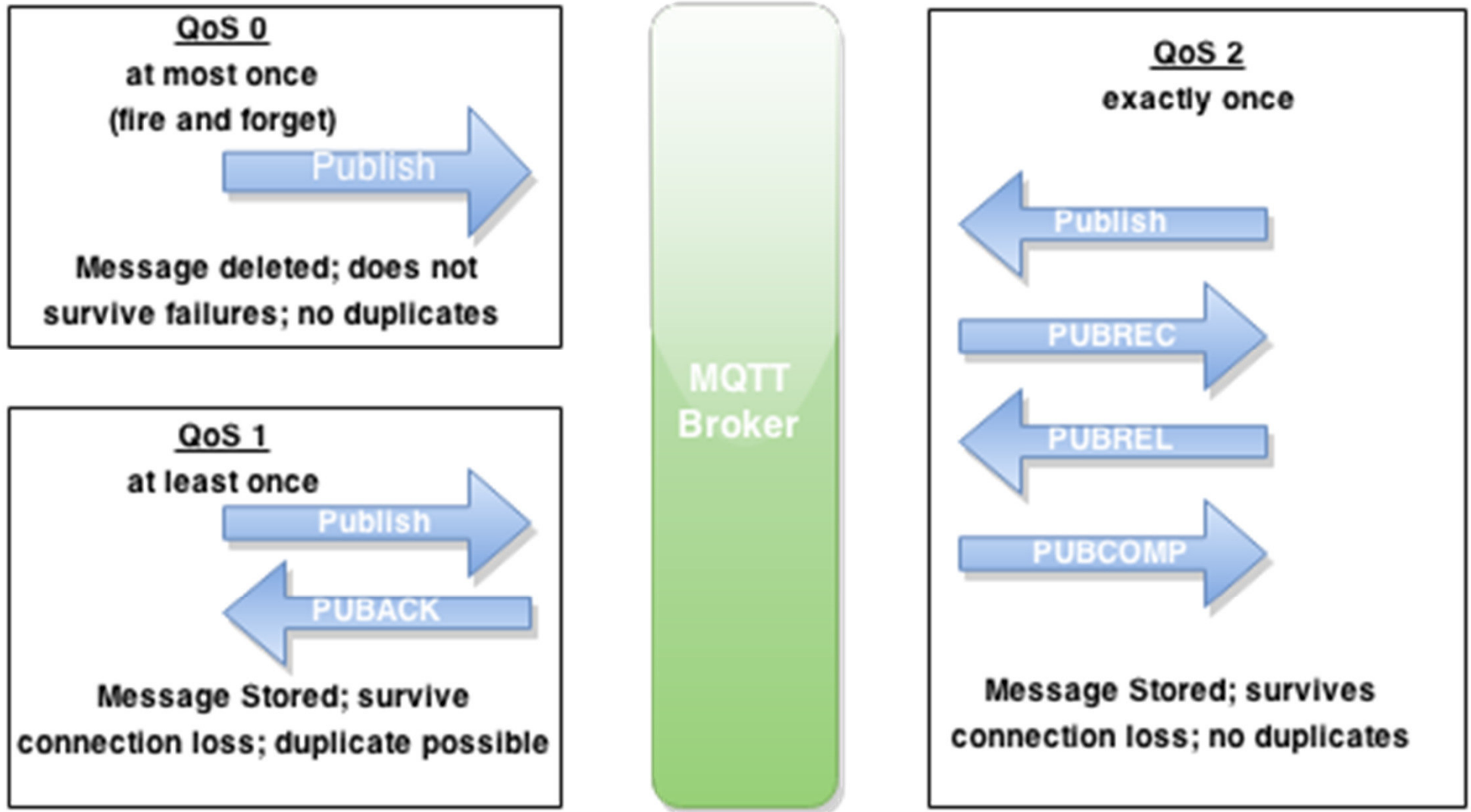
At Most
Once
(QoS=0)

At Least
Once
(QoS=1)

Exactly
Once
(QoS=2)

MQTT Quality of Service Levels

Message Queue Telemetry Transport (MQTT) Quality of Service (QoS)



Simple Text Orientated Messaging Protocol (STOMP) 1.2 (2012)

Einfaches, standardisiertes textbasiertes Protokoll zur freien Verwendung

```
client-command      = "SEND"  
                    | "SUBSCRIBE"  
                    | "UNSUBSCRIBE"  
                    | "BEGIN"  
                    | "COMMIT"  
                    | "ABORT"  
                    | "ACK"  
                    | "NACK"  
                    | "DISCONNECT"  
                    | "CONNECT"  
                    | "STOMP,,
```

```
server-command      = "CONNECTED"  
                    | "MESSAGE"  
                    | "RECEIPT"  
                    | "ERROR"
```

Stomp 

Agenda

Kommunikation verändert sich	Mobiles Messaging – Möglichkeiten & Heraus- forderungen	Messaging- Protokolle – MQTT & STOMP	Messaging- Produkte für MQTT & STOMP	Beispiel (JavaScript, Java, CLI)	Fazit – wird möglich
	folgenden Möglichkeiten	STOMP	STOMP	JavaScript	

Eclipse Paho 1.1 und Eclipse Mosquitto 1.4

- **Paho-Projekt** <http://www.eclipse.org/paho> seit 2012 bei Eclipse
- **Ziel:** Implementierungen von Standard-Messaging-Client-Protokollen im Machine-to-Machine-Bereich (**M2M**) Clients für *Java, C, C++, JavaScript, Lua, Python, Go Client, C# .Net* und *WinRT*
- **Mosquitto-Server:** entstand 2010 als Open-Source-Implementierung des Internet-of-Things-Protokolls **MQTT 3.1.1**

```
mosquitto.conf
mosquitto.dll
mosquitto.exe
mosquitto.top
mosquitto_passwd.exe
mosquitto_pub.exe
mosquitto_sub.exe

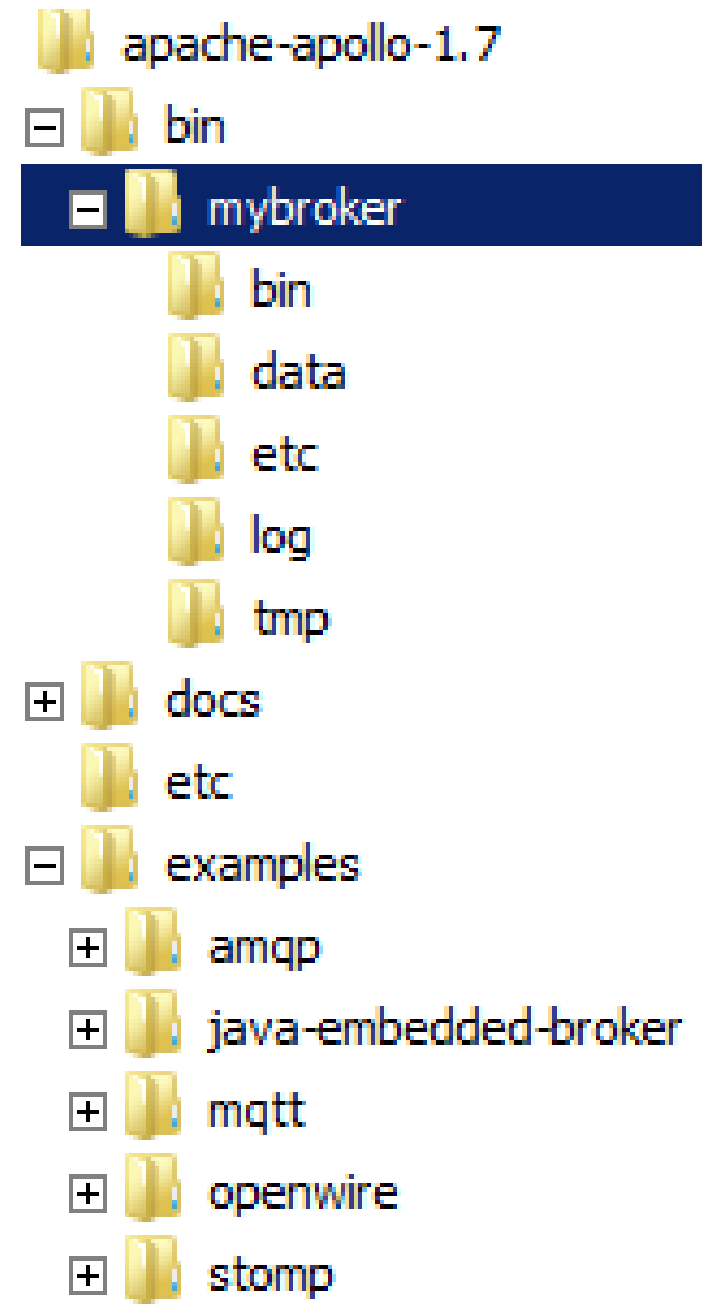
D:\mosquitto14>
```

Latest release: 1.1	Downloads
C client for Windows/Unix/Mac	1.0.3
Java client and utilities	1.0.2
Android service	1.0.2
Python client	1.1
JavaScript client	1.0.1
C/C++ MQTT Embedded clients	1.0.0
.Net and WinRT client (M2Mqtt)	4.0.0.0



Apache Apollo 1.7.1

- Apache **ActiveMQ** Nachfolger
- **Connectoren**
 - TCP
 - UDP
 - WebSocket
- **Protokolle**
 - **STOMP**
 - **MQTT**
 - OpenWire
 - AMQP

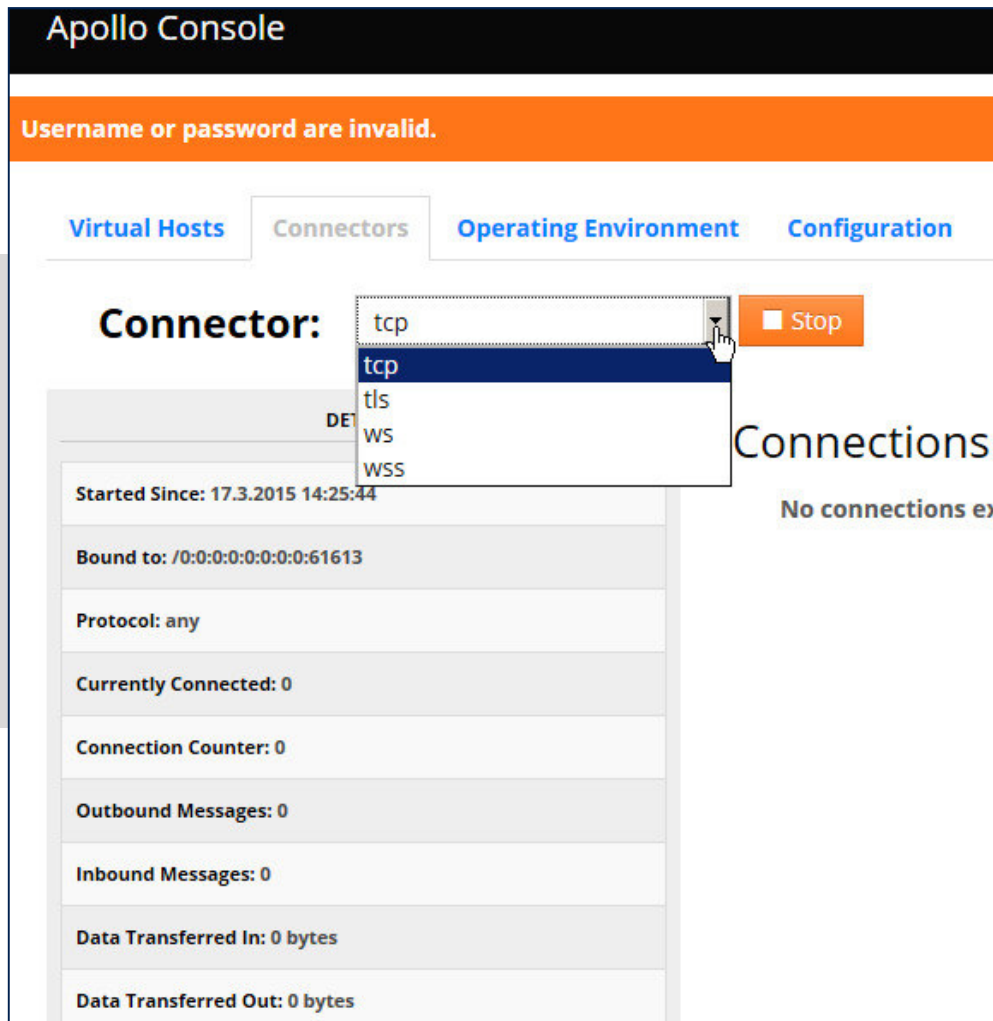


Apollo Broker apollo.xml

```
{APOLLO_HOME}/bin/apollo create mybroker  
..\mybroker\bin\apollo-broker run  
curl -u "admin:password" http://localhost:61680/broker  
curl -u "admin:password" http://localhost:61680/broker.json  
curl -u "admin:password" \  
'http://localhost:61680/broker/connections.json'
```

```
Loading configuration file 'D:\apache-apollo-1.7\mybroker\etc\apollo.xml'.  
INFO | OS      : Windows 7 6.1  
INFO | JVM      : Java HotSpot(TM) 64-Bit Server VM 1.7.0_72 (Oracle Corporation)  
INFO | Apollo   : 1.7 (at: D:\apache-apollo-1.7)  
INFO | Starting store: leveldb store at D:\apache-apollo-1.7\mybroker\data  
INFO | Accepting connections at: tcp://0.0.0.0:61613  
INFO | Accepting connections at: tls://0.0.0.0:61614  
INFO | Accepting connections at: ws://0.0.0.0:61623/  
INFO | Accepting connections at: wss://0.0.0.0:61624/  
INFO | virtual host startup is waiting on store startup  
INFO | virtual host startup is no longer waiting. It waited a total of 1 seconds.  
INFO | Administration interface available at: https://127.0.0.1:61681/  
INFO | Administration interface available at: http://127.0.0.1:61680/
```

Apollo Webconsole



apollo.xml

- <http://127.0.0.1:61680>
- <https://127.0.0.1:61681>

users.properties

groups.properties

admin/password

Agenda



Apollo STOMP WebSocket JavaScript-Anwendung

```
<script src="js/stomp.js"></script>
```

```
<script>//</pre></div><div data-bbox="41 186 324 209" data-label="Text"><pre>$(document).ready(function() {</pre></div><div data-bbox="59 212 268 233" data-label="Text"><pre>  if(window.WebSocket) {</pre></div><div data-bbox="77 238 304 260" data-label="Text"><pre>    var client, destination;</pre></div><div data-bbox="77 264 436 286" data-label="Text"><pre>    $('#connect_form').submit(function() {</pre></div><div data-bbox="96 290 416 312" data-label="Text"><pre>      var url = $('#connect_url').val();</pre></div><div data-bbox="96 315 454 339" data-label="Text"><pre>      var login = $('#connect_login').val();</pre></div><div data-bbox="96 342 508 365" data-label="Text"><pre>      var passcode = $('#connect_passcode').val();</pre></div><div data-bbox="96 367 454 390" data-label="Text"><pre>      destination = $('#destination').val();</pre></div><div data-bbox="96 393 351 416" data-label="Text"><pre>      client = <b>Stomp.client(url)</b>;</pre></div><div data-bbox="96 419 380 442" data-label="Text"><pre>      client.debug = function(str) {</pre></div><div data-bbox="114 445 407 468" data-label="Text"><pre>        $('#debug').append(str + "\n");</pre></div><div data-bbox="96 472 119 494" data-label="Text"><pre>      };</pre></div><div data-bbox="96 500 556 522" data-label="Text"><pre>      <b>client.connect(login, passcode, function(frame) {</b></pre></div><div data-bbox="114 525 444 547" data-label="Text"><pre>        client.debug("connected to Stomp");</pre></div><div data-bbox="114 551 528 573" data-label="Text"><pre>        $('#connect').fadeOut({ duration: 'fast' });</pre></div><div data-bbox="114 577 351 599" data-label="Text"><pre>        $('#connected').fadeIn();</pre></div><div data-bbox="114 602 574 625" data-label="Text"><pre>        client.subscribe(destination, function(message) {</pre></div><div data-bbox="133 628 648 651" data-label="Text"><pre>          $('#messages').append("&lt;p&gt;" + message.body + "&lt;/p&gt;\n");</pre></div><div data-bbox="114 654 147 676" data-label="Text"><pre>        });</pre></div><div data-bbox="96 681 128 703" data-label="Text"><pre>      });</pre></div><div data-bbox="96 707 220 728" data-label="Text"><pre>      return false;</pre></div><div data-bbox="77 733 109 755" data-label="Text"><pre>    });</pre></div><div data-bbox="12 784 343 807" data-label="Text"><pre>$('#send_form').submit(function() {</pre></div><div data-bbox="41 810 407 834" data-label="Text"><pre>  var text = $('#send_form_input').val();</pre></div><div data-bbox="41 837 147 859" data-label="Text"><pre>  if (text) {</pre></div><div data-bbox="59 862 388 886" data-label="Text"><pre>    <b>client.send(destination, {}, text)</b>;</pre></div><div data-bbox="59 889 341 912" data-label="Text"><pre>    $('#send_form_input').val("");</pre></div><div data-bbox="41 915 54 937" data-label="Text"><pre>  }</pre></div><div data-bbox="12 941 239 964" data-label="Text"><pre>return false;      });</pre></div><div data-bbox="25 961 137 979" data-label="Text"><pre>© Materna GmbH 2015</pre></div><div data-bbox="467 125 777 150" data-label="Text"><p>file:///D:/apache-apollo-1.7/examples/stomp/websocket/index.html</p></div><div data-bbox="720 164 993 189" data-label="Section-Header"><h2>Apollo STOMP WebSocket Chat Example</h2></div><div data-bbox="477 221 587 248" data-label="Section-Header"><h3>Server Login</h3></div><div data-bbox="505 290 682 309" data-label="Text"><p>WebSocket URL <input type="text" value="ws://localhost:61623"/></p></div><div data-bbox="554 336 624 354" data-label="Text"><p>User <input type="text" value="admin"/></p></div><div data-bbox="532 382 642 399" data-label="Text"><p>Password <input type="password" value="....."/></p></div><div data-bbox="527 426 674 445" data-label="Text"><p>Destination <input type="text" value="/topic/chat.general"/></p></div><div data-bbox="597 495 646 513" data-label="Text"><p><b>Connect</b></p></div><div data-bbox="455 961 539 979" data-label="Page-Footer"><p>www.materna.de</p></div><div data-bbox="955 961 979 980" data-label="Page-Footer"><p>38</p></div>
```

Apollo STOMP WebSocket Chat Anwendung

Apollo STOMP WebSocket Chat Example

Chat Room

Hello World

Hello World

Send

Disconnect

Debug Log

```
Opening Web Socket...
Web Socket Opened...
>>> CONNECT
accept-version:1.1,1.0
heart-beat:10000,10000
login:admin
passcode:password

<<< CONNECTED
version:1.1
server:apache-apollo/1.7
host-id:mybroker
session:mybroker-4
heart-beat:100,10000
user-id:admin

connected to server apache-apollo/1.7
send PING every 10000ms
check PONG every 10000ms
connected to Stomp
>>> SUBSCRIBE
id:sub-0
destination:/topic/chat.general

>>> PING
<<< PONG
>>> SEND
destination:/topic/chat.general
content-length:11

Hello World
<<< MESSAGE
subscription:sub-0
message-id:mybroker-41
destination:/topic/chat.general
content-length:11

Hello World

>>> PING
```

[Virtual Hosts](#) [Connectors](#) [Operating Environment](#) [Configuration](#)

Connector:

WS Stop

DETAILS

Started Since: 2.9.2015 08:36:22

Bound to: /0.0.0.0:61623

Protocol: any

Currently Connected: 2

Connection Counter: 4

Outbound Messages: 9

Inbound Messages: 5

Connections

Page 1 of 1:

☐	Remote Address	Protocol	User	Data In	Data Out	Msgs In	Msgs out
☐	@/127.0.0.1:1481	mqtt		148 bytes	181 bytes	2	6
☐	@/127.0.0.1:1631	stomp	admin	142 bytes	124 bytes	0	0

Apollo STOMP-Java-Anwendung

```
String user = env("APOLLO_USER", "admin");
String password = env("APOLLO_PASSWORD", "password");
String host = env("APOLLO_HOST", "localhost");
int port = Integer.parseInt(env("APOLLO_PORT", "61613"));
String destination = arg(args, 0, "/topic/event");

StompJmsConnectionFactory factory = new StompJmsConnectionFactory();
factory.setBrokerURI("tcp://" + host + ":" + port);

Connection connection = factory.createConnection(user, password);
connection.start();
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
Destination dest = new StompJmsDestination(destination);
MessageProducer producer = session.createProducer(dest);
producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

for( int i=1; i <= messages; i ++ ) {
    TextMessage msg = session.createTextMessage(body);
    msg.setIntProperty("id", i);
    producer.send(msg);
    if( (i % 1000) == 0 ) {
        System.out.println(String.format("Sent %d messages", i));
    }
}

producer.send(session.createTextMessage("SHUTDOWN"));
connection.close();
```

CLI-CRUD mit MQTT-Mosquitto-Server tcp://iot.eclipse.org:1883

- Über **HTTP-to-MQTT Bridge**

```
curl -X PUT --data-binary "Hello World"
```

```
http://eclipse.mqttbridge.com/JS curl
```

```
http://eclipse.mqttbridge.com/JS
```

```
curl -X DELETE http://eclipse.mqttbridge.com/JS
```

- Mit Mosquitto-Client Thema JS abonnieren

```
mosquitto_sub -h iot.eclipse.org -p 1883 -t JS -v
```

- Nachrichten an ein bestimmtes Thema senden

```
mosquitto_pub -h iot.eclipse.org -p 1883 -r -t JS -m  
"Hello World"
```

- Etwas komfortabler geht das mit der Eclipse RCP-Anwendung MQTT Client Tools

Eclipse Paho MQTT Client und MQTT-Mosquitto-Server

tcp://iot.eclipse.org:1883

The screenshot displays the Eclipse Paho MQTT Client interface, which is divided into several panels. The top-left panel, titled 'MQTT Options', contains the 'Connection' section with fields for 'Server URI' (tcp://iot.eclipse.org:1883), 'Client ID' (apollo-19418010071195), and 'Status' (Connected). Below this is the 'Subscription' section, which includes a table for managing subscriptions. The table has columns for 'Topic' and 'QoS', and a checkbox for 'Retained'. The first row shows a subscription for 'JS' with 'QoS 0 - At Most Once'. The bottom-left panel, titled 'Publication', contains fields for 'Topic' (JS), 'QoS' (0 - At Most Once), and 'Message' (Hello World3). The top-right panel, titled 'History', shows a table of recent events. The bottom-right panel, titled 'MQTT Options', contains settings for 'Enable SSL', 'Enable HA', 'Enable LWT', 'Keep Alive', 'Connection Timeout', 'Enable Login', 'Enable Persistence', 'SSL Settings', 'High Availability (HA)', and 'Last Will and Testament (LWT)'. The 'Enable SSL' section is expanded, showing fields for 'Key Store Location', 'Key Store Password', 'Trust Store Location', and 'Trust Store Password'. The 'High Availability (HA)' section is also expanded, showing a list of 'Server URIs' (tcp://localhost:1883, ssl://localhost:8883). The 'Last Will and Testament (LWT)' section is expanded, showing fields for 'Topic' (lwt), 'QoS' (0 - At Most Once), and 'Message'.

Event	Topic	Message	QoS	Retained
Connected				
Subscribed	JS		0	
Received	JS	Hello World2	0	Yes
Published	JS	Hello World3	0	No
Received	JS	Hello World3	0	No

Topic	QoS	Retained
JS	0 - At Most Once	☐

Topic	QoS	Retained
lwt	0 - At Most Once	☐

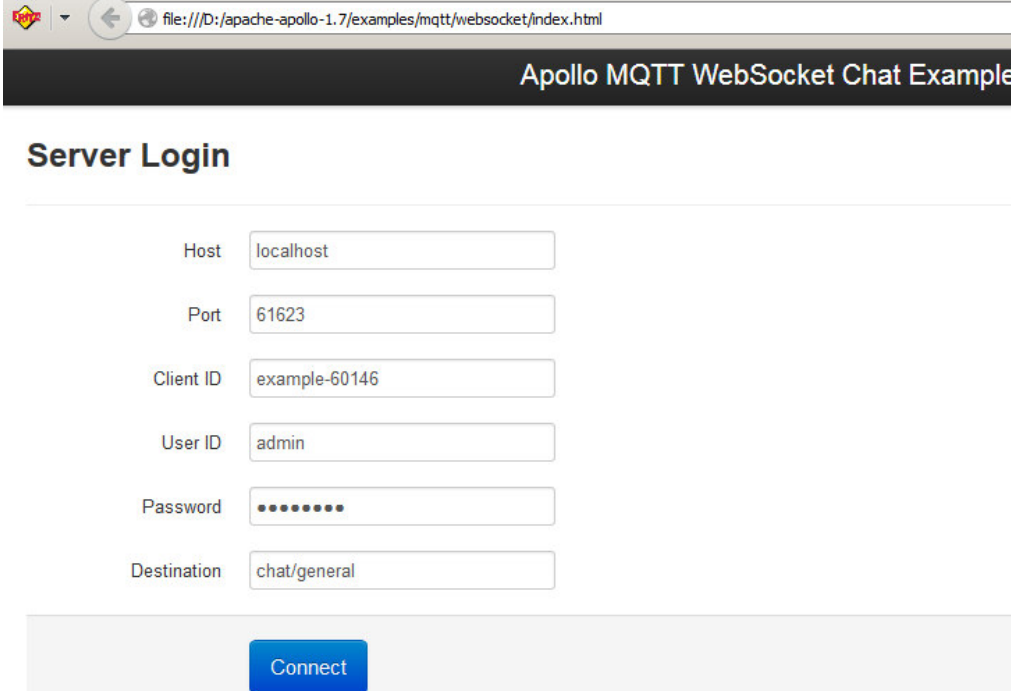
Eclipse Paho MQTT Java-Client

```
String topic          = "JS";
String content        = "Hello World";
int qos              = 0;
String broker        = "tcp://iot.eclipse.org:1883";
String clientId       = "JSSample";
MemoryPersistence persistence = new MemoryPersistence();
MqttClient sampleClient = new
MqttClient(broker, clientId, persistence);
MqttConnectOptions connOpts = new MqttConnectOptions();
connOpts.setCleanSession(true);
System.out.println("Connecting to broker: " + broker);
sampleClient.connect(connOpts);
System.out.println("Connected");
System.out.println("Publishing message: " + content);
MqttMessage message = new MqttMessage(content.getBytes());
message.setQos(qos);
sampleClient.publish(topic, message);
System.out.println("Message " + content + " published");
sampleClient.disconnect();
```

Apollo MQTT-Java-/JavaScript-Anwendung

apache-apollo-1.7\examples\mqtt\
java\src\main\java\example

- Listener.java
 - Publisher.java
- websocket\index.html
- js/mqttws31.js



The screenshot shows a web browser window with the address bar displaying 'file:///D:/apache-apollo-1.7/examples/mqtt/websocket/index.html'. The page title is 'Apollo MQTT WebSocket Chat Example'. Below the title, there is a section titled 'Server Login'. This section contains a form with the following fields: 'Host' (value: localhost), 'Port' (value: 61623), 'Client ID' (value: example-60146), 'User ID' (value: admin), 'Password' (masked with dots), and 'Destination' (value: chat/general). At the bottom of the form, there is a blue 'Connect' button.

Agenda

Kommunikation
verändert sich

Mobiles
Messaging –
Möglichkeiten &
Heraus-
forderungen

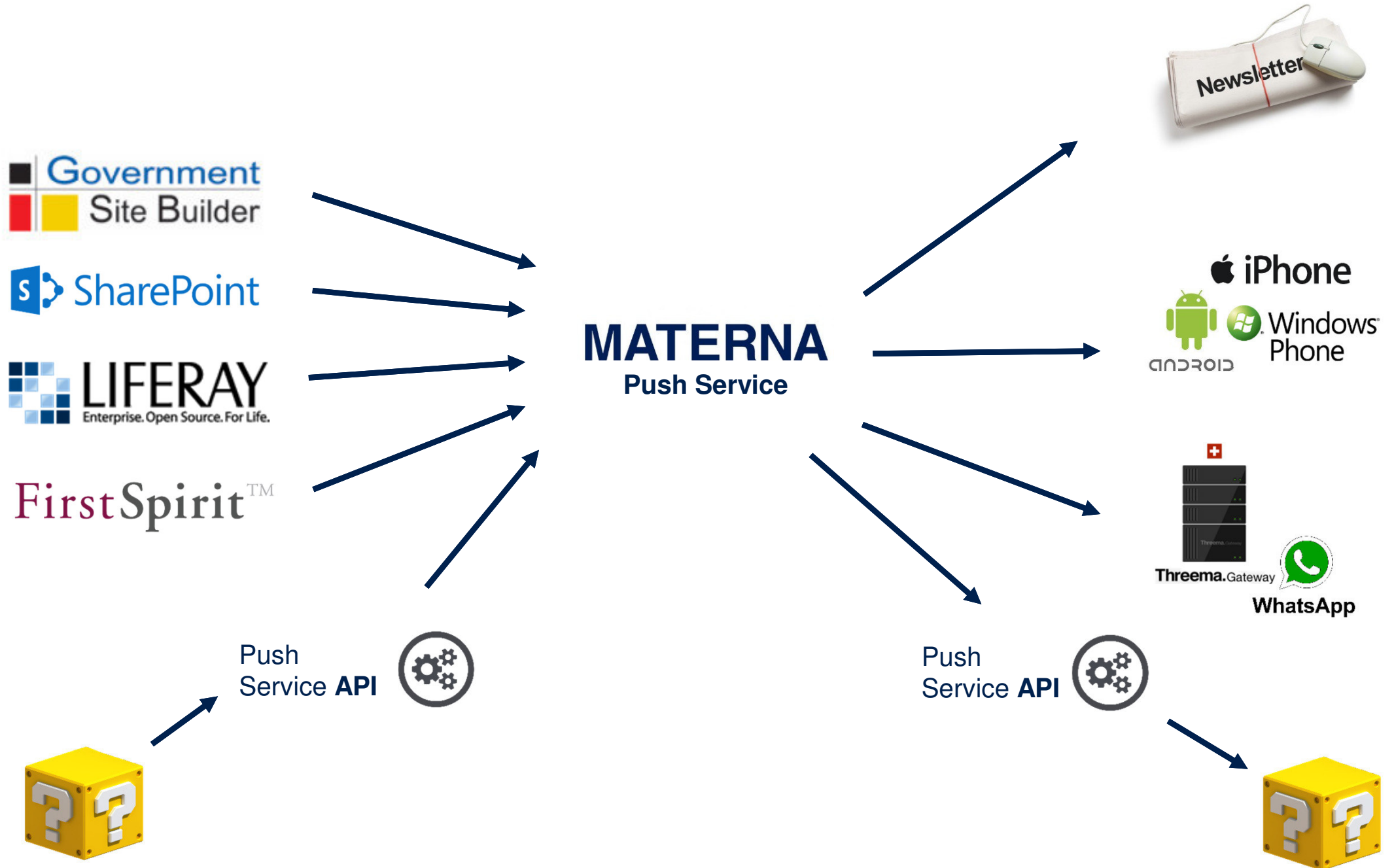
Messaging-
Protokolle –
MQTT &
STOMP

Messaging-
Produkte für
MQTT &
STOMP

Beispiel
(JavaScript,
Java, CLI)

Fazit –
wird möglich

Unified Push Service for CMS

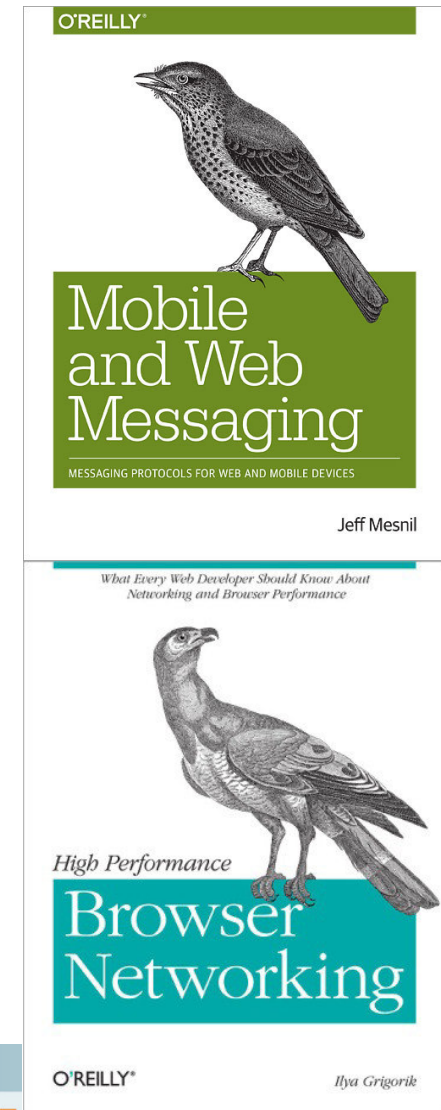


Fazit

- 
- A person wearing a dark suit jacket over a light blue shirt is holding a white rectangular card. The card contains a bulleted list of five items. The person's hand is visible at the top of the card, and their face is partially visible in the background, out of focus.
- Asynchrone, offline Kommunikation nimmt zu (IoT, M2M)
 - Polyglottes Messaging ist möglich, aber noch jung!
 - MQTT/STOMP/XMPP bieten breite Unterstützung
 - Notification kann Kundenbindung & Nutzung erhöhen (UX)
 - Das Beste aus beiden Welten kombinieren

Links

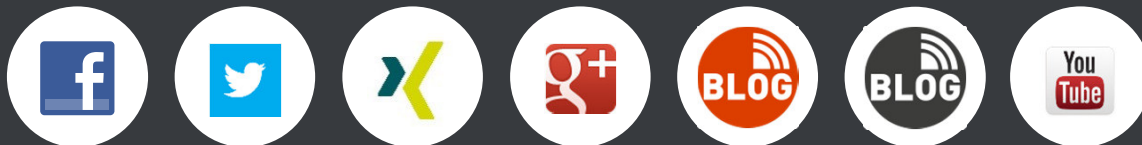
- STOMP Protokoll-Spezifikationen <http://stomp.github.io>
- MQTT-Organisation <http://mqtt.org>
- MQTT-Spec <http://docs.oasis-open.org/mqtt/mqtt/v3.1.1>
- MQTT.js <https://www.npmjs.com/package/mqtt>
- Public MQTT Brokers
https://github.com/mqtt/mqtt.github.io/wiki/public_brokers
- Eclipse Paho <http://eclipse.org/paho>
- Mosquitto-Server <http://mosquitto.org>
- Apache Apollo <https://activemq.apache.org/apollo>
- Google Cloud Messaging (GCM) Server
<https://developer.android.com/google/gcm/server.html>
- <https://dzone.com/refcardz/getting-started-with-mqtt>
- AeroGear <http://aerogear.org>



Fragen?



Vernetzt.



Kontakt.

Materna GmbH
Frank Pientka
Voßkuhle 37
44141 Dortmund
+49 231 5599-8854
Frank.Pientka@materna.de