

Implementation of a OneAPI-REST interface

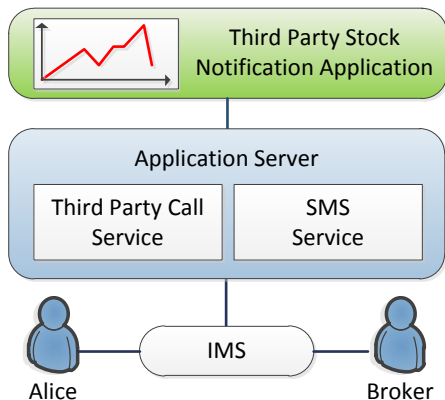
for integrating web services in an
IMS-based telecommunication network

[Helge Reelfs](#)

1. Motivation
2. Environment
3. Architecture
4. Example
5. Evaluation
6. Conclusion

- Telecommunication network cores nowadays merge to all-IP networks
- Telecommunication services
- Very powerful end-user devices
- Web 2.0 Apps
- **Merge both domains!**
- TODO:
 - Create telecommunication services
 - Create interface between these and the web

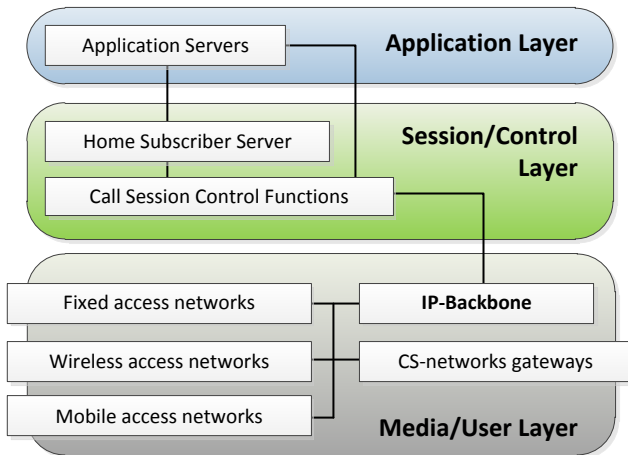
Example: 3rd Party Service composition



Third Party App:

1. Stock drops under value Y
2. $\rightarrow$ `ThirdPartyCall(Alice, Broker)`
3. $\leftarrow$ `busy(Alice)`
4. $\rightarrow$ `SendSMS(Alice, stock + value)`

IP Multimedia Subsystem



SOAP/XML for Web Services + REST Interface

Services

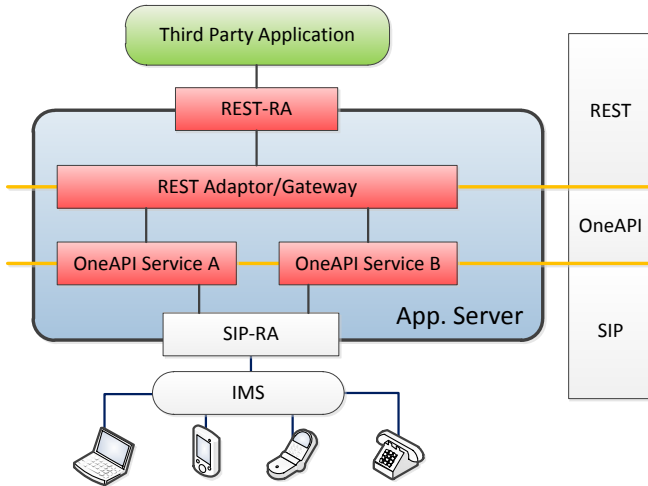
v1.0	SMS, MMS Location Payment	published
v2.0	Click-to-call In-app billing Remaining Credits lookup Data-Connection-Profile Call-notification	published
v3.0	Quality of Service Application Triggering Video Streaming	

- Representational State Transfer
- Resources
- HTTP for transport
- Addressing via URI
- Use HTTP methods directly
 - POST
 - PUT
 - GET
 - DELETE
- Data is directly put into HTTP body

→ Low protocol overhead

- Event Driven Platform
 - No active execution threads
 - Event method handler & fire methods
- Purpose: Telecommunication applications
- Reusable logical blocks
 - Service Building Blocks
- Internal communication channels
 - Activities
- Boundary between SLEE and external world
 - Resource Adaptors

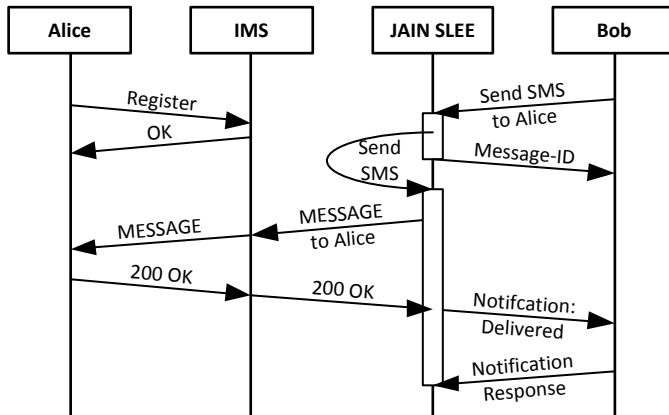
Interface and Application Architecture



Short Messaging Service

- Send
 - Send SMS
 - Request delivery status
- Receive
 - Get SMS for a subscription
 - Delete received SMS
- Notifications
 - SMS delivery notifications
 - SMS receipt notifications
 - Manage subscriptions

Example: Send an SMS

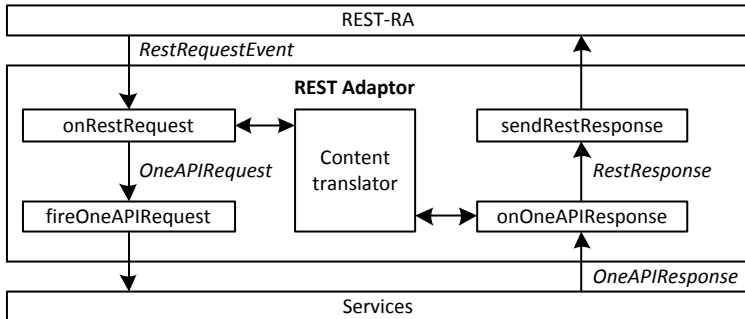


OneAPI Send SMS REST Request

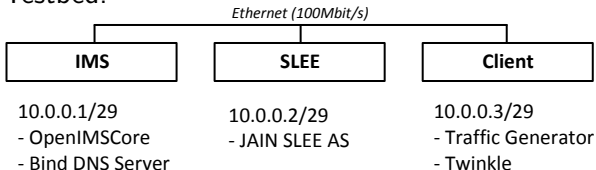
```
POST /1/smsmessaging/outbound/sip%3ahelge%40example.de/requests HTTP/1.1
Host: example.com:80
Content-Type: application/xml; charset=UTF-8
Accept: application/xml,application/json
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<sms:outboundSMSMessageRequest xmlns:sms="urn:oma:xml:rest:sms:1" >
  <address>sip:destination@foo.com</address>
  <senderAddress>sip:helge@example.de</senderAddress>
  <senderName>Helge</senderName>
  <receiptRequest>
    <notifyURL>http://example.de/DeliveryInfoNotification</notifyURL>
  </receiptRequest>
  <outboundSMSTextMessage>
    <message>
      A little man stands in the woods, quiet and mute.
      He wears a purple little coat...
    </message>
  </outboundSMSTextMessage>
  <clientCorrelator>someCorrelatorString</clientCorrelator>
</sms:outboundSMSMessageRequest>
```

Implementation Details: Incoming request



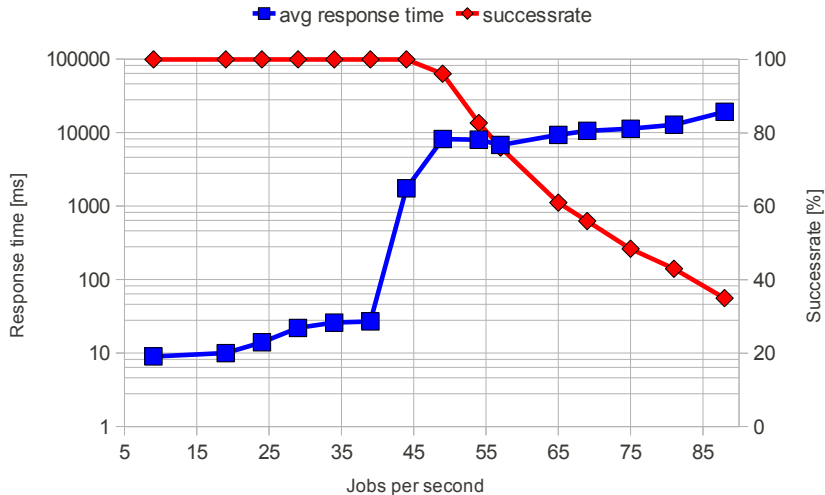
- Testbed:



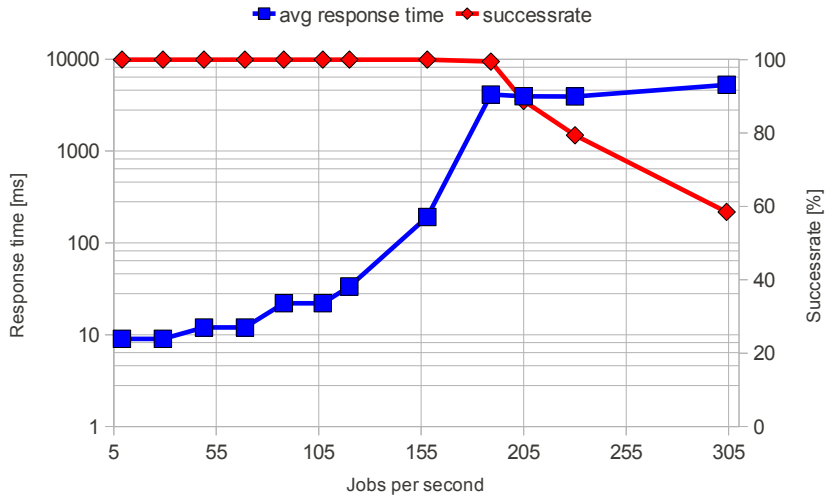
- Tests (selection):

- Send and deliver SMS
→ complete vertical communication
- SMS invocation
→ interface

Send and deliver SMS



SMS invocation



- Heavy load causes higher losses
 - Late tests revealed that sockets remain in TCP `CLOSE_WAIT` state and do not get closed properly
 - Not all connections can be accepted anymore
- Need for more investigation on the network part of the RA
- Overall performance seems quite good
- Low response times as long as there are no losses

- Telecommunication networks merge to all-IP networks
- High demand of more sophisticated services
- Hype around “apps” and Web 2.0 applications
- Combining both domains seems obvious
- OneAPI defines key services
 - They enable more abstract service compositions
- JAIN SLEE AS is meant for these applications
- Running prototype performs quite good

- Implement more OneAPI Services on this basis
- Efficient Service Composition
- Multimedia-Services (OneAPI v3.0)