INTELLIGENCE AS A SERVICE. SELF-MANAGEMENT OF SERVICES
Intelligent infrastructure design for the IoT (DII)
(Part One)

UNIVERSIDAD COMPLUTENSE MADRID

- Alfonso González Briones
- Based on material created by Jorge Gómez Sanz

Introduction to the subject

- **Remote Procedure Call**
- **Introduction SaaS and Serverless**
- **Introduction PaaS and IaaS**

Remote Procedure Call

- **REST or WS**
  - Makes a difference
  - Hamad, H., Saad, M., & Abed, R. (2010). Performance Evaluation of RESTful Web Services for Mobile Devices. Int. Arab J. e-Technol., 1(3), 72-78.

Number of array elements	Message Size (byte)				Time (Milliseconds)			
	SOAP/HTTP		REST (HTTP)		SOAP/HTTP		REST (HTTP)	
	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition
2	351	357	39	32	781	781	359	359
3	371	383	48	36	828	781	344	407
4	395	409	63	35	828	922	359	375
5	418	435	76	39	969	1016	360	359
6	443	461	93	43	875	953	359	359
7	465	487	104	47	875	875	469	360
8	493	513	127	51	984	875	437	344

Service response time (milliseconds) and message size (bytes) of the concatenation string.

Remote Procedure Call

- **A Complex Decision - Decision Overview**

Architectural Decision and AAs	REST	WS-*
Integration Style	1 AA	2 AAs
Shared Database		
File Transfer		
Remote Procedure Call	✓	✓
Messaging		✓
Contract Design	1 AA	2 AAs
Contract-first		✓
Contract-last		✓
Contract-less	✓	
Resource Identification	1 AA	n/a
Do-it-yourself	✓	
URI Design	2 AA	n/a
"Nice" URI scheme	✓	
No URI scheme	✓	
Resource Interaction Semantics	2 AAs	n/a
Lo-REST (POST, GET only)	✓	
Hi-REST (4 verbs)	✓	
Resource Relationships	1 AA	n/a
Do-it-yourself	✓	
Data Representation/Modeling	1 AA	1 AA
XML Schema	(✓)[a]	✓
Do-it-yourself	✓	
Message Exchange Patterns	1 AA	2 AAs
Request-Response	✓	✓
One-Way		✓
Service Operations Enumeration	n/a	≥3 AAs
By functional domain		✓
By non-functional properties and QoS		✓
By organizational criterion (versioning)		✓
Total Number of Decisions, AAs	**8**, 10	**5**, ≥10

[a]Optional

Table 2: Conceptual Comparison Summary

Architectural Decision and AAs	REST	WS-*
Transport Protocol	1 AA	≥7 AAs
HTTP	✓	✓[a]
waka [13]	(✓)[b]	
TCP		✓
SMTP		✓
JMS		✓
MQ		✓
BEEP		✓
IIOP		✓
Payload Format	≥6 AAs	1 AA
XML (SOAP)	✓	✓
XML (POX)	✓	
XML (RSS)	✓	
JSON [10]	✓	
YAML	✓	
MIME	✓	
Service Identification	1 AA	2 AA
URI	✓	✓
WS-Addressing		✓
Service Description	3 AAs	2 AAs
Textual Documentation	✓	
XML Schema	(✓)[c]	✓
WSDL	✓[d]	✓
WADL [18]	✓	
Reliability	1 AA	4 AAs
HTTPR [38][e]	(✓)	(✓)
WS-Reliability		✓
WS-ReliableMessaging		✓
Native		✓
Do-it-yourself	✓	✓
Security	1 AA	2 AAs
HTTPS	✓	✓
WS-Security		✓
Transactions	1 AA	3 AAs
WS-AT, WS-BA		✓
WS-CAF		✓
Do-it-yourself	✓	✓
Service Composition	2 AAs	2 AAs
WS-BPEL		✓
Mashups	✓	
Do-it-yourself	✓	✓
Service Discovery	1 AAs	2 AAs
UDDI		✓
Do-it-yourself	✓	✓
Implementation Technology	many	many
...	✓	✓
Total Number of Decisions, AAs	**10**, ≥17	**10**, ≥25

[a]Limited to only the verb POST
[b]Still under development
[c]Optional
[d]WSDL 2.0
[e]Not standard

Table 3: Technology Comparison Summary

Architectural Principle and Aspects	REST	WS-*
Protocol Layering	yes	yes
HTTP as application-level protocol	✓	
HTTP as transport-level protocol		✓
Dealing with Heterogeneity	yes	yes
Browser Wars	✓	
Enterprise Computing Middleware		✓
Loose Coupling, aspects covered	yes, 2	yes, 3
Time/Availability		✓
Location (Dynamic Late Binding)	(✓)	✓
Service Evolution:		
Uniform Interface	✓	
XML Extensibility	✓	✓
Total Principles Supported	**3**	**3**

Table 1: Principles Comparison Summary

Remote Procedure Call

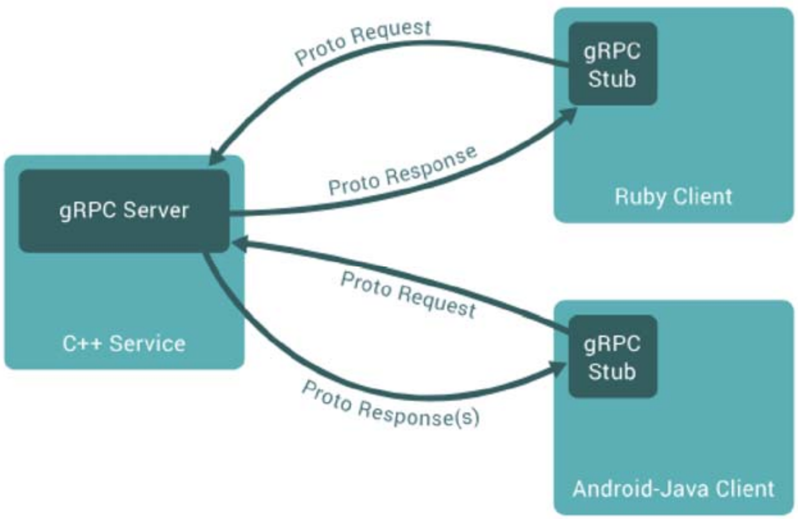
- **So far, they have seen web services and multi-tier architecture, and microservices.**
  - RPC implies that one program calls another program on another machine:
    - asynchronous
    - synchronous
  - It is different from a remote socket invocation in that sockets offer only two operations: read and write.
  - RPC makes it possible to provide a remote API. Google RPC is an implementation of this approach.
  - There are others, such as Java RMI or Apache Thrift https://thrift.apache.org/.

Remote Procedure Call

- **Questions that arise…**
  - Do both programs have to be written in the same language ==> polyglot
  - What does the other program do when it is not invoked?
  - What happens if two or more computers want to invoke the program

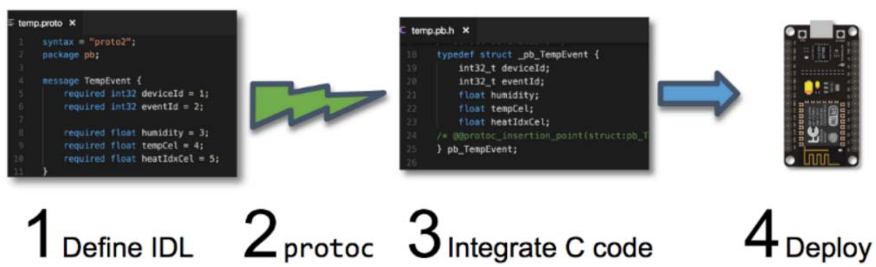
Remote Procedure Call

- **Google RPC**
  - A description of RPC elements



Remote Procedure Call

- **Google RPC & IoT**
  - The main reason for integration is the relationship between backend and edge computing. Efficiency
  - ESP8266 Programming
  - But you can't directly use gRPC on an arduino!!!!
    - However, it can handle data streams: https://www.youtube.com/watch?v=c9z_o5lu0dI



gRPC - http://grpc.io

Efficient IoT with the ESP8266, Protocol Buffers, Grafana, Go, and Kubernetes - https://medium.com/grpc/efficient-iot-with-the-esp8266-protocol-buffers-grafana-go-and-kubernetes-a2ae214dbd29

Remote Procedure Call

- **gRPC Vs WS**
  - SOA also offers us an API / gRPC
    - WS requires a heavy infrastructure (JBoss, Tomcat) / gRPC does not.
    - WS may define the protocol / gRPC may not
    - WS can globally organize how different services are coordinated / gRPC cannot.
  - WS cannot give uniform data types across the /gRPC infrastructure.
  - Interesting new ideas: bidirectional streaming in gRPC
  - A summary of why REST "*sucks*" when compared to gRPC: https://www.youtube.com/watch?v=RoXT_Rkg8LA

Remote Procedure Call

- **Benchmark REST Vs gRPC**
  - Workload performance:
    - Normal loading: REST > gRPC => string concatenation
    - Heavy load: REST < gRPC => small numbers, big numbers, small and big sentences (??)
  - Each language has a different performance:

Remote Procedure Call

- **gRPC IoT**
  - Benchmark (Research Gate):
    - shorturl.at/mIPR4
  - More Works (google Scholar):
    - shorturl.at/ftKP8

Remote Procedure Call

- **gRPC IoT - energy consumption (client side)**
  - LocaL: a smartphone Samsung S5 (ARM 2.1Gz Octacore)
  - Remote: a windows 7 server (Intel Core 2 Duo 2.2Ghz)

INPUT SIZE	INPUT TYPE	ALGORITHM	CASE	COMPLEXITY	LOCAL	REST	SOAP	SOCKET	GRPC
1000	Int	Bubble Sort	Best	$\mathcal{O}(n)$	**0,0066**	0,0971	0,1957	0,4731	0,1535
1000	Int	Heap Sort	Worst	$\mathcal{O}(n \log n)$	**0,0084**	0,0979	0,1281	0,2462	0,1173
1000	Int	Heap Sort	Best	$\mathcal{O}(n \log n)$	**0,0120**	0,1105	0,1749	0,2819	0,2023
1000	Int	Selection Sort	Worst	$\mathcal{O}(n^2)$	**0,0144**	0,0808	0,1116	0,2373	0,1345
1000	Int	Bubble Sort	Worst	$\mathcal{O}(n^2)$	**0,0273**	0,1455	0,1888	0,1995	0,1032
1000	Int	Selection Sort	Best	$\mathcal{O}(n^2)$	**0,0286**	0,1073	0,1723	0,3590	0,2247
10000	Int	Bubble Sort	Best	$\mathcal{O}(n)$	**0,0562**	0,3842	5,2375	0,5723	0,6103
1000	Float	Heap Sort	Worst	$\mathcal{O}(n \log n)$	**0,0705**	0,1379	0,2245	0,2734	0,1442
1000	Float	Bubble Sort	Best	$\mathcal{O}(n)$	**0,0788**	0,1221	0,1796	0,2532	0,0909
1000	Float	Heap Sort	Best	$\mathcal{O}(n \log n)$	**0,0846**	0,1164	0,1504	0,2004	0,1635
10000	Int	Heap Sort	Worst	$\mathcal{O}(n \log n)$	**0,0852**	0,5927	0,6151	0,6963	1,0327
1000	Float	Selection Sort	Worst	$\mathcal{O}(n^2)$	**0,0859**	0,1101	0,1304	0,2528	0,1669
1000	Float	Selection Sort	Best	$\mathcal{O}(n^2)$	**0,0908**	0,1225	0,1044	0,3539	0,1566
10000	Int	Heap Sort	Best	$\mathcal{O}(n \log n)$	**0,0989**	0,5654	0,6684	0,5826	1,5551
1000	Float	Bubble Sort	Worst	$\mathcal{O}(n^2)$	**0,1035**	0,1103	0,1515	0,3035	0,1214

- Small vectors

Chamas, C. L., Cordeiro, D., & Eler, M. M. (2017, November). Comparing REST, SOAP, Socket and gRPC in computation offloading of mobile applications: An energy cost analysis. In Communications (LATINCOM), 2017 IEEE 9th Latin-American Conference on (pp. 1-6). IEEE.

Remote Procedure Call

- **gRPC IoT - energy consumption (client side)**
  - LocaL: a smartphone Samsung S5 (ARM 2.1Gz Octacore)
  - Remote: a windows 7 server (Intel Core 2 Duo 2.2Ghz)

INPUT SIZE	INPUT TYPE	ALGORITHM	CASE	COMPLEXITY	LOCAL	REST	SOAP	SOCKET	GRPC
100000	Float	Heap Sort	Best	$\mathcal{O}(n \log n)$	6,9552	7,9610	12,2928	**5,5922**	10,8711
10000	Object	Selection Sort	Worst	$\mathcal{O}(n^2)$	44,3543	**5,7710**	15,4622	10,0284	10,8408
100000	Float	Heap Sort	Worst	$\mathcal{O}(n \log n)$	8,4738	10,3493	7,4396	**6,1408**	12,7246
100000	Float	Bubble Sort	Best	$\mathcal{O}(n)$	**6,3832**	10,3493	7,3783	6,6799	15,2624
10000	Object	Bubble Sort	Worst	$\mathcal{O}(n^2)$	84,3696	**9,7146**	15,6492	10,9943	15,6132
100000	Int	Selection Sort	Best	$\mathcal{O}(n^2)$	66,9138	13,4757	16,9476	**11,9876**	39,6920
100000	Float	Selection Sort	Best	$\mathcal{O}(n^2)$	121,2813	**16,0041**	20,2135	22,1771	33,9588
100000	Object	Bubble Sort	Best	$\mathcal{O}(n)$	**20,0052**	70,5636	104,8919	242,5625	166,3286
100000	Object	Heap Sort	Best	$\mathcal{O}(n \log n)$	**22,3046**	44,8823	133,8276	77,6200	86,2444
100000	Float	Selection Sort	Worst	$\mathcal{O}(n^2)$	194,0500	31,8115	31,3616	**22,9645**	35,6874
100000	Int	Selection Sort	Worst	$\mathcal{O}(n^2)$	88,2045	29,5133	**23,3795**	34,6518	30,5247
100000	Object	Heap Sort	Worst	$\mathcal{O}(n \log n)$	**23,6646**	199,2946	129,3667	57,9254	77,6200
100000	Int	Bubble Sort	Worst	$\mathcal{O}(n^2)$	242,5625	47,9136	**36,2710**	46,1987	56,5979
100000	Float	Bubble Sort	Worst	$\mathcal{O}(n^2)$	388,1000	50,7320	51,7467	**37,6796**	86,2444
100000	Object	Selection Sort	Best	$\mathcal{O}(n^2)$	-	**215,6111**	251,9835	242,5625	232,86
100000	Object	Selection Sort	Worst	$\mathcal{O}(n^2)$	-	**250,3870**	388,1	275,9824	277,2143
100000	Object	Bubble Sort	Worst	$\mathcal{O}(n^2)$	-	**388,1000**	**388,1000**	**388,1000**	**388,1000**

- Large vectors

Chamas, C. L., Cordeiro, D., & Eler, M. M. (2017, November). Comparing REST, SOAP, Socket and gRPC in computation offloading of mobile applications: An energy cost analysis. In Communications (LATINCOM), 2017 IEEE 9th Latin-American Conference on (pp. 1-6). IEEE.

Remote Procedure Call

- **To experience**
  - Clone: https://github.com/escalope/examplegrpc
  - Execute the following instructions. First example of a simple client server.
    - $ mvn verify
    - $ # Run the server in one terminal
    - $ mvn exec:java -Dexec.mainClass=io.grpc.examples.helloworld.HelloWorldServer
    - $ # In another terminal run the client
    - $ mvn exec:java -Dexec.mainClass=io.grpc.examples.helloworld.HelloWorldClient
  - Now a streaming client to the server:
    - $ mvn verify
    - $ # Run the server in one terminal
    - $ mvn exec:java -Dexec.mainClass=io.grpc.examples.manualflowcontrol.ManualFlowControlServer
    - $ # In another terminal run the client
    - $ mvn exec:java -Dexec.mainClass=io.grpc.examples.manualflowcontrol.ManualFlowControlClient

Remote Procedure Call

- **Conclusions**
  - Integration between the different languages is possible.
  - WS/RPC/REST are not the fastest/productive ones either.
  - RPC requirements vs. WS requirements vs. REST requirements
  - RPC: needs a daemon (a light one) WS: needs an application server (heavy)
  - REst: need a light server