

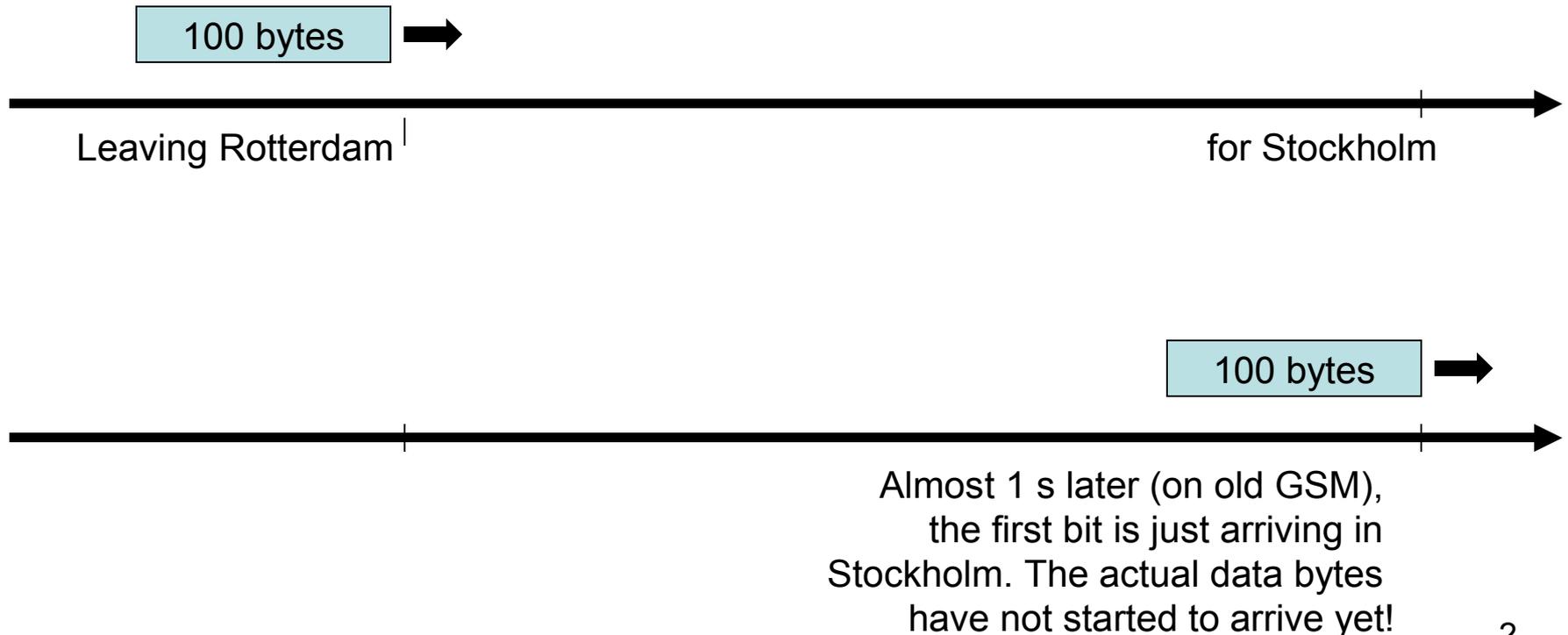
Cloud Symposium
Rotterdam 2009-10-22 - 23

Latency Threatens the SOA Cloud

Sven-Håkan Olsson

Now, what is latency?

- Latency: A delay before something actually takes place
- Well, delay sounds bad, but does that give any problem – broadband performance has increased enormously!
- Yes, bandwidth has increased, but latency is still a problem!



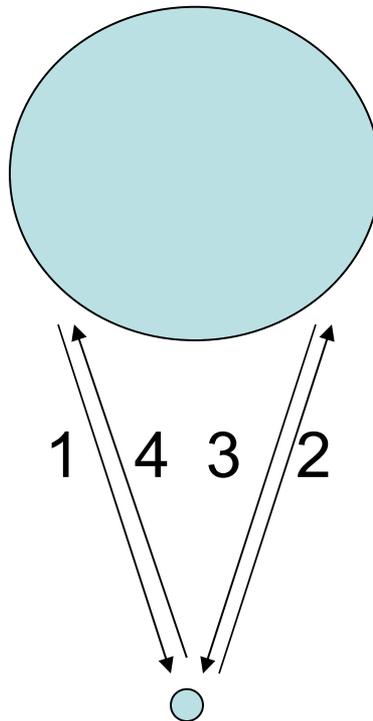
But latency can get improved a lot, right?

- Yes, improvements are seen. GSM data traffic may perhaps get as low as 0,1 s on a short distance, only counting the GSM part itself. But it is difficult to get yet better!
- There is a definitive ceiling: Speed of light! Einstein stuff... $3 * 10^8$ m/s in vacuum. 70% of that, in cable.
- To get around the globe takes more than 0,13 s. Impossible to improve!
- To this we have to add all sorts of other latencies...
- And by the way, where was that cloud server you were using? Australia? Taiwan?



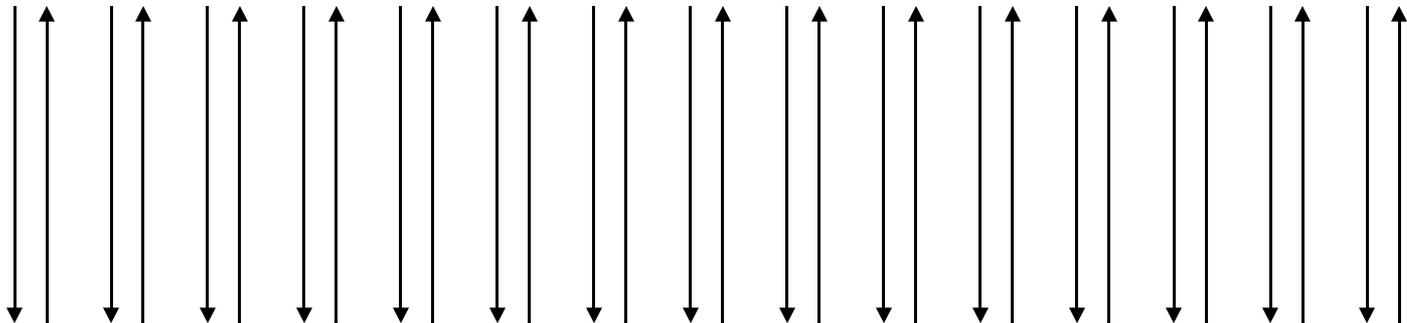
Via geostationary satellite?

- To stay in the same spot on the sky the satellite has to be at a distance of 72 000 km
- Time for a request+response Rotterdam-satellite-Stockholm and back (round-trip-time, RTT) is 0,5 s!
- Satellites not so common today for IT, but a good example

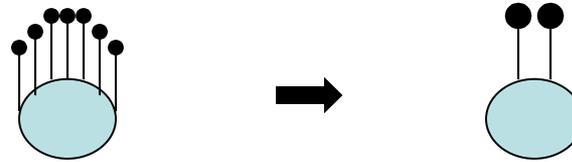


The sewing machine anti-pattern

- Typical of old 2-tier client/server solutions but this problem appears in recent development too.
- A lot of small remote calls, sequentially. Chatty.
- Latency adds up!
- 40 calls (not uncommon for old c/s) to the other side of the world to show a form, consumes over 5 s without even counting the actual data transfer time!
- Classic web site is NOT like this (parallel fetching). Works fine to Taiwan! But Ajax used the wrong way may get chatty before displaying complete result...



Naïve SOA



- An example: A common OO design is to use SET and GET methods for every separate little attribute.
- It is incredibly easy to generate a SOAP interface directly from that class and say that it is a SOA interface.
- The ultimate sewing machine anti-pattern! Try to call a majority of the set methods in that class from here to a Cloud server in Taipei. Or Redmond, maybe you don't quite know in the Cloud...
- XML is perfect for grouping together hierarchical data – use that and create coarse granular SOA interfaces instead. (Useful for other reasons as well.) Reduces latency impact enormously because of fewer round-trips!

Different types of clouds

- In my session 13:15 today I categorized Cloud types.
- Type 4, sometimes called ~PaaS, gives the possibility to easily deploy your own app code into the Cloud servers.
- For example in Google App Engine or Microsoft Azure you could theoretically deploy just a small class. With one click (almost) in Visual Studio the class gets deployed in the Cloud. Maybe a bit too easy for inexperienced developers 😊 If you don't think "coarse granular" you'll end up with terrible latency effects, as in the previous slide!
- So, SOA combined with the Cloud definitively should mean coarse granular interface calls.

Wait, this has all been about SYNC!

- All my examples until now has been "calls", RPC, synchronous access etc.
- The caller **WAITS** until the response gets back from the Cloud at the other side of the globe, **THEN** issues the next call or whatever. Latencies add up.
- Reasons for talking only about sync so far:
 - It is really common in real-life, also in SOA
 - Good-old-SOAP access IS really sync
 - Even REST really IS sync
 - Client/server guys tend to think sync
 - Many solutions "near the UI" gets to be sync
 - Above all: The business requirements may lead to sync!

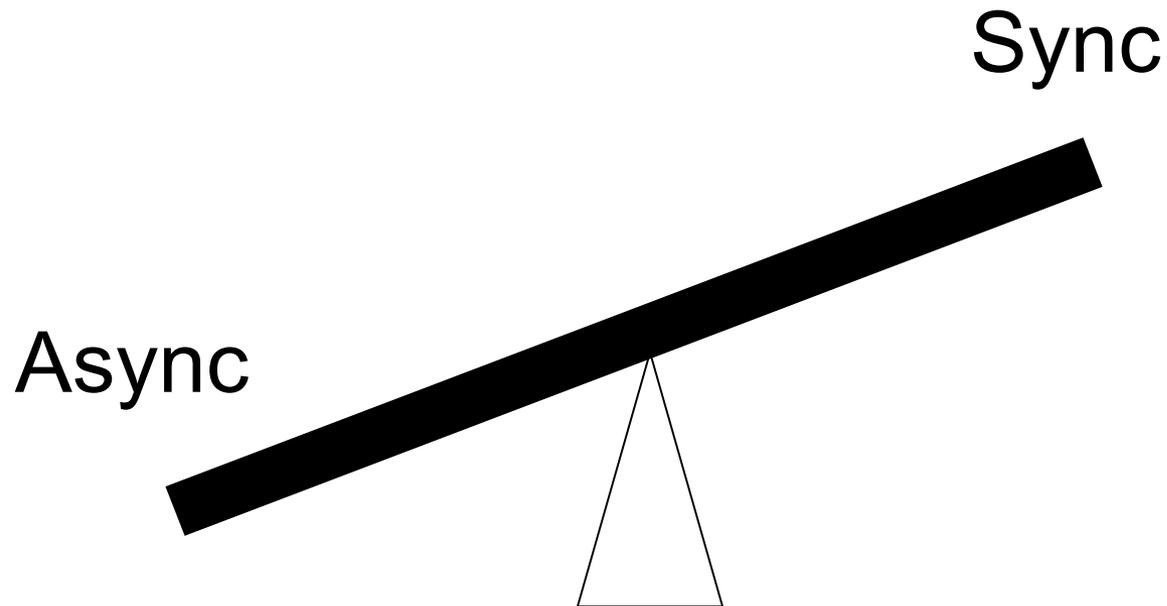
Maybe sync is an anti-pattern? ASYNC!

- Just send the data, DON'T wait until the server has processed it, just continue.
- So, asynchronous instead!
- Usually avoids latency problems
- Reduces dependency on that the other server is up just now – higher total uptime probability
- Other positive aspects of the mythical "loose coupling"...

- Many say that ALL proper SOA interfaces should be async
- Typical example just now: MS Azure coding examples, almost all async.

- Watch out so that the logic doesn't require you to wait for an async response before sending the next async request and so on – this is AS BAD as sync!

It's not that simple, both async and sync have pro et contra points – you have to balance



Async, pro et contra:

- + Avoids latency problems
- + Loose coupling in the "time domain"
- + Suitable for message oriented patterns and replication
- + Suitable for EDA – Event Driven Architecture

- - Requires que software:
ESB, EAI, MOM, WS-RM, homemade...
- - Complicated, expensive and error-prone (!)
error/exception handling
 - Exception logic dispersed in time and place.
 - Both business exceptions and technical exceptions must be fixed

- - Above all: The business requirements may lead to sync!

- Price request

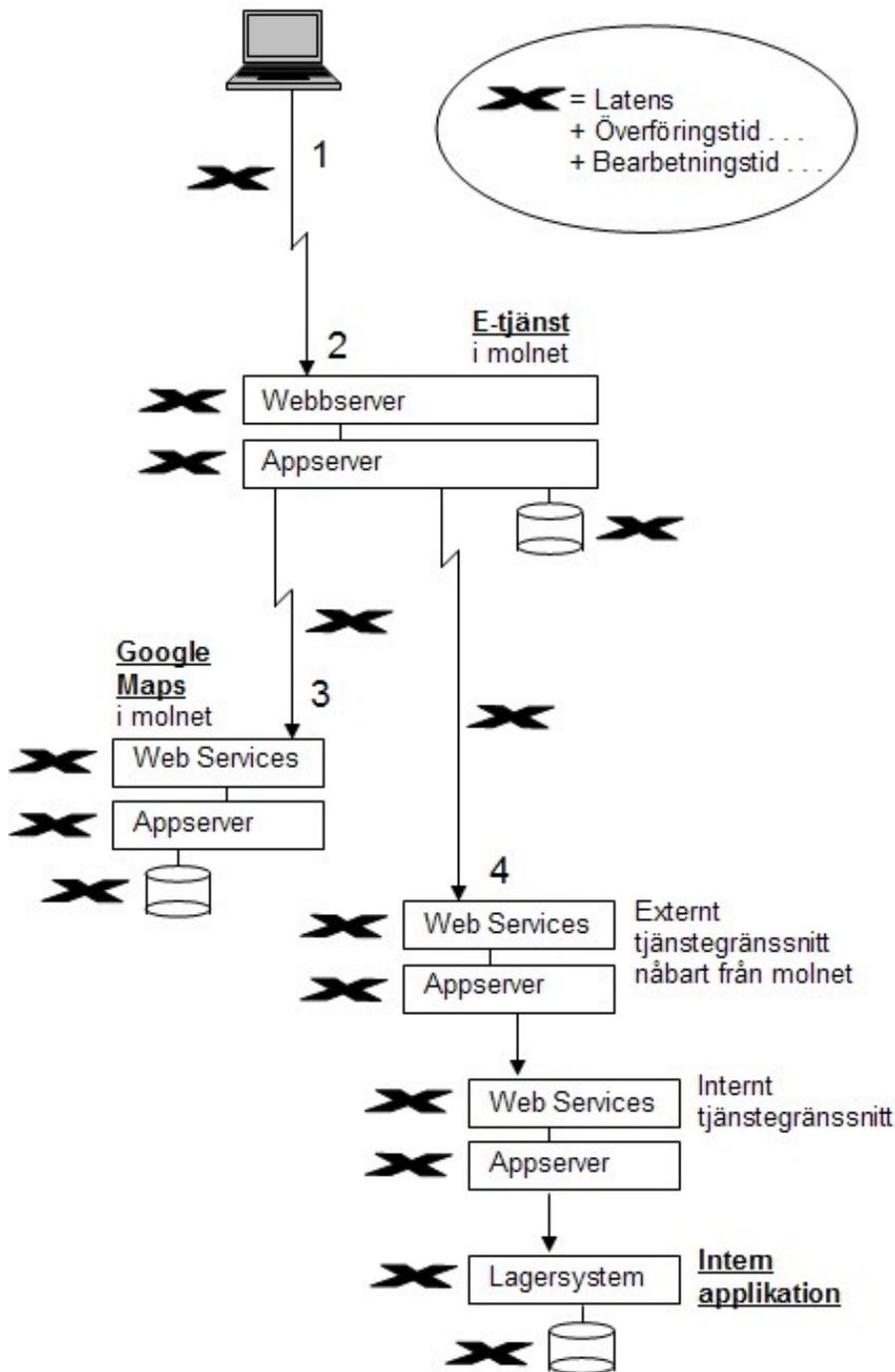
- Usually no problem at all to request against a replicated price register – prices often can be stated never to change during the day
- Replication uses an async pattern
- No latency problems, nor uptime dependencies

- Request for stock availability

- Can I accept this order, have we got the item in stock?
- This request may be necessary to run sync towards a central service!
- Have to handle latency optimization, and uptime dependencies

- Invoice is to be sent to customer

- Usually not at all necessary to be fast. The same day usually sufficient
- Async, batch transfer etc
- No latency problems, nor uptime dependencies



A complete example

- A cloud SOA service is to reach other services
- A lot of latencies, in many of the stages (shown with: **X**)
- Not only network latency! Also app server latency, DOM latency, proxy latency, multiple encapsulation layers, etc etc
- So, coarse granular!

Conclusion

- Cloud servers may be located anywhere in the world
- Even if you know where they are, it might be far
- Latency is very difficult to decrease. Also a theoretical minimum with respect to certain distance.
- Cannot use naïve SOA interfaces for the Cloud when possibly long distance (or other reasons for latency)
 - Coarse granular instead!
 - Async patterns when possible!
- Sometimes you have to balance, sync may be ok.

Sven-Håkan Olsson is an independent consultant, course leader and speaker who focuses on application architecture and SOA. Since 1977 he has worked in a large number of IT development projects, ranging from embedded microcontrollers to main-frames. He has carried out modeling, architecture design and programming in diverse business areas. He has also specialized in reviews of problem projects.

Sven-Håkan Olsson holds an MScEE from the Royal Institute of Technology in Stockholm. In May 2008, he was appointed one of the 'top developers in Sweden' by the magazine IDG Computer Sweden. He is also a co-founder of the consultancy company Know IT.



Please feel free to contact
me if you have comments
or suggestions!

D·E·F·I·N·I·T·I·V·U·S

Tel +46 708 840134

sven-hakan.olsson@definitivus.se

www.definitivus.se