Extending the Enterprise to Mobiles

(and how to reliably push data to mobiles)

John Pagonis
Developer Services

Extending the Enterprise to Mobiles

- Enterprise on the go? Why?
- Typical enterprise use cases
- How to push data from the back end to mobiles
- Bidirectional comms to mobiles
  - … The enemy within
  - … How to do it
  - … What's been done
- Rich clients vs. browser based scenarios on the go

Enterprise on the go? Why?

- Real time decision making (info at hand and in time)
- Harnessing of context
- Local decision making
- Global distribution of knowledge
- Latency reduction (time critical scenarios)
- Business process optimisation
- Usability vs. Laptops (responsiveness, battery etc)
- Ubiquity
- …and so much more

Enterprise on the go? Why?

Which all lead to

- Increased responsiveness
- Increased competitiveness
- Trimming the fat from operations
- More knowledge

and in general

increasing efficiency and the bottom line

What can enterprises do? (step by step)

- Mobilise
  - … Remotely pulling info
  - … CRM
- Extend
  - … Operations extend to mobiles
  - … power of up to date info for field staff becomes available
- Push
  - … time critical information to mobile workforce
  - … knowledge captured in the back-ends goes out
- Capture & Distribute
  - … Distribution of information as it happens
  - … real time data acquisition

Typical enterprise use cases

- CRM
- Sales Force Automation
- Field Service Automation
- Data Acquisition
- Enterprise Resource Planning
- Bla bla bla bla

  All of the above have natural extensions to mobiles and can benefit from harnessing context ! – location, pictures, audio, video, human memory, time etc

So what's the problem then? (enough with the marketing:-)

Pick one: Mobilise, extend, push, capture & distribute

All of the above have

- operational
- cultural
- technical difficulties
- business
- Ethical, privacy,security

When push comes to shove…..

Push is the most difficult one and the most important

Why is it difficult:

- In most cases devices are behind NAT
- With no direct inbound connections allowed
- Connection tracking in NATs is problematic
- Roaming – IP addressing issues
- No coverage
- SMS delivery is best effort
- MMS (uses SMS btw for notifications)

NATs and Mobs….

(STUN RFC 3489 has a very good treatment of what goes on, in the twisted world of NATs.)

- No NAT spec or standard really!!
- Operator NATs are configured differently
- NATs have limited resources, therefore connection tracking suffers and records get timed-out
- While your session is still valid!!

The enemy within (the network that is)

- You cannot have incoming GPRS/UMTS connections (unless you're the operator or have special arrangements)
- Even the connection that you have running can (silently) fail
- Without your client being notified synchronously, that is….because the NAT decided so
- Device IP address may need to change if roaming

How to push data from the back end to mobiles

- Deal with the operator - really
  - … Can get IP netblock, APNs, incoming connections etc
- Out of band messaging
  - … SMS based
  - … but it is unreliable
- Polling
  - … DO NOT…I REPEAT DO NOT
- Use a VPN (or mobile IP) setup
  - … not easy to deploy and run
- Maintain a TCP based session
  - … And do in band notifications from the mothership

Push to mobiles: the way to do it (using TCP)

- Create a TCP session from the mobile
- Since this is bidirectional :-) push data from the ‘’server’’ to the mobile – or just notify the mobile to fetch (like IMAP-IDLE does)
- Let it be

….and now the fun starts

Reliable push to mobiles….it's fun

….and now the fun starts

- You need an application layer protocol of course
- You need to make sure that the operator NAT hasn't forgotten all about your session
- You need to check that the connection is still on
- Both the TCP server and clients need those checks
  - … imagine what happens when the client realises the session is unusable but the server doesn't, while the client reconnects and tries to authenticate
- Your application protocol needs to make sure that messages are not lost (on that layer)
- Application protocol needs to ensure that messages are stored, forwarded and retransmitted (according to app needs – once or more?)

Keeping alive… ah ah ah keeping alive…

From RFC 1122:

> Implementors MAY include "keep-alives" in their TCP implementations, although this practice is not universally accepted. If keep-alives are included, the application MUST be able to turn them on or off for each TCP connection, and they MUST default to off.

> Keep-alive packets MUST only be sent when no data or acknowledgement packets have been received for the connection within an interval. This interval MUST be configurable and MUST default to no less than two hours.

From RFC 1122 (continued)

The TCP specification does not include a keep-alive mechanism because it could:

(1) cause perfectly good connections to break during transient Internet failures

(2) consume unnecessary bandwidth ("if no one is using the connection, who cares if it is still good?"); and

(3) cost money for an Internet path that charges for packets.

For those that didn’t get it!

- TCP sessions stay connected by default without wasting bandwidth (modulo error conditions)
- Determining whether a peer is alive should be a task of the application/middleware and thus be an application (protocol) layer responsibility
- Only if you run into the danger of keeping a server in the ESTABLISHED state forever and thus using up its resources, you should switch on keep-alive probes

…and even if we did, it wouldn’t help our scenarios much anyway

Finding out your session is a zombie

Only one true way:

When you write to the socket, you'll know

Because of NATs, unfortunately you need to cater in the application protocol for zombie sessions

Based on the usage scenario design your probes, especially when waiting for a request completion!

Different applications will have different requirements

Doing so will create a better user experience

What's been done

- Nokia TCP based proxy/registar for incoming connections to mobile Apache
- RIM's operator-based solution
- My-Channels' Nirvana (both TCP and HTTP based)
- Lampdesk's LampSync
- Push email solutions
  - … IMAP-IDLE, Intellisync, Visto, ActiveSync, Seven, RIM etc
- SyncML with OOB signalling (OMA DS/DM)

- Roll your own, on XMPP :-) , have a look at SIP proxies as well, etc

Rich clients vs. browser based scenarios

- There is a lot of context and intelligence captured on the client that deals with the issues discussed
- Browsers are good for user initiated pulling of info
- Pushing demands an on device agent to communicate autonomously and asynchronously
- What happens to a browser scenario when the radio connection is lost?
- What can a browser do when the NAT records expire
  - … Did the transaction complete?
  - … Did the data arrive?
  - … Can you retry?

Thank You ☺

Questions ?

Coming soon:

"Symbian OS Communications Programming - New Edition“, by Iain Campbell