Mobile Computing
#MC04 Push/Notifications

CS60002: Distributed Systems
Winter 2006-2007

Update to “parts we will discuss”

- ✔ Device databases
  - Flash, OR/direct
- ✔ Synchronization
  - Algorithms
- ➔ Push/notifications
  - Scale to MM
- ✗ Handheld design
  - CPU, RTOS, battery
- ✗ Core Mobile Apps
  - Email/IM, PDA,browse
- ✗ IP Protocols
  - IMS, SIMPLE/XMPP
- • Broadcast
  - Algorithms
- • Device Management
  - Software & Config

Push/Notifications

- What is it?
- Brief history of implementations
  - Identify three paradigms
- Implementation challenges
- Analysis of some implementations
  - Scalability of the three paradigms
  - Slides inadequate. Please take notes.

What is “Push”?

- Client data updated asynchronously
  - No user action needed
  - Server or peer “pushes” an update to the client
- Older than computing itself
  - Telegraph, phone, TV
- Instant Messaging
  - Talk (PDP-11 1970s, Unix 1980s), IRC (1988)
  - ICQ (1996), AOL IM (1997)
  - Jabber/XMPP (2000)

Push reinvented

- TCP/HTTP Server Push
  - Pointcast.com (1992)
  - `<META HTTP-EQUIV="Refresh" CONTENT=15>`
  - `multipart/x-mixed-replace` (Netscape 1995)
  - Microsoft channels (1996?)
- Push e-mail and messaging
  - IMAP IDLE (RFC 2177 – 1997)
  - BlackBerry (1999)
  - WAP Push (WAP 1.2 – 2001)

Push reinvention (contd.)

- Push email (contd.)
  - Good, Seven, Visto, Funambol (2004 ..)
  - “Direct Push” (Windows Mobile 2006)
  - iPhone Push email (Summer 2008)
- IP telephony
  - SIP (1999) => SIMPLE (2002)
  - XMPP (1998) => Google Jingle (2008)

Push reinvention (contd.)

- TCP/HTTP Server Push – Take 2
  - Comet or reverse-AJAX (2007?)
  - More to come?

Customer demand is the mother of reinvention!

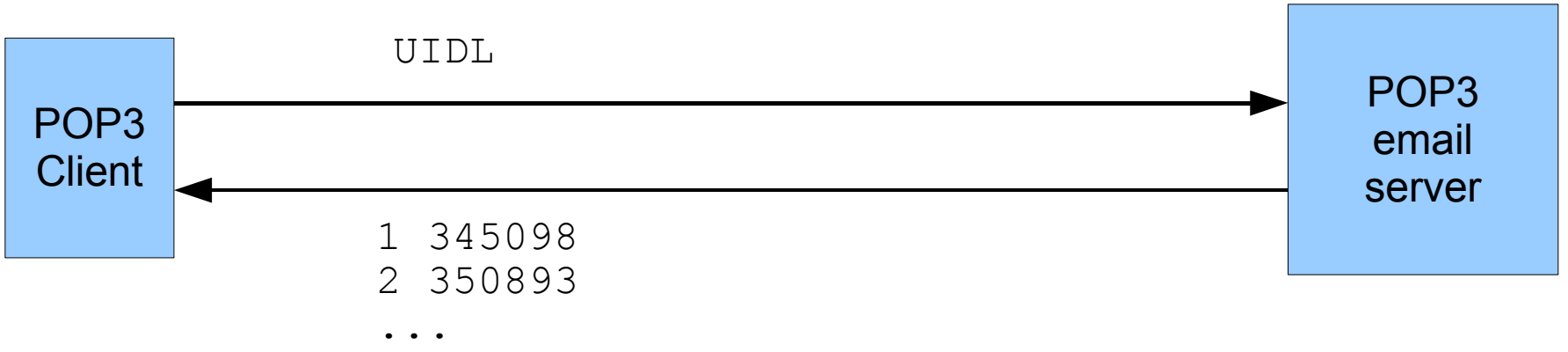
Three paradigms

1) Polling
   - Client polls server frequently
   - Variant: Asynchronous exchanges
2) Client listens on an inbound port
   - Peers and Server post to that port
   - Variant: Poke-n-pull
3) Client stays connected to a server
   - All peers post to that server

Many challenges

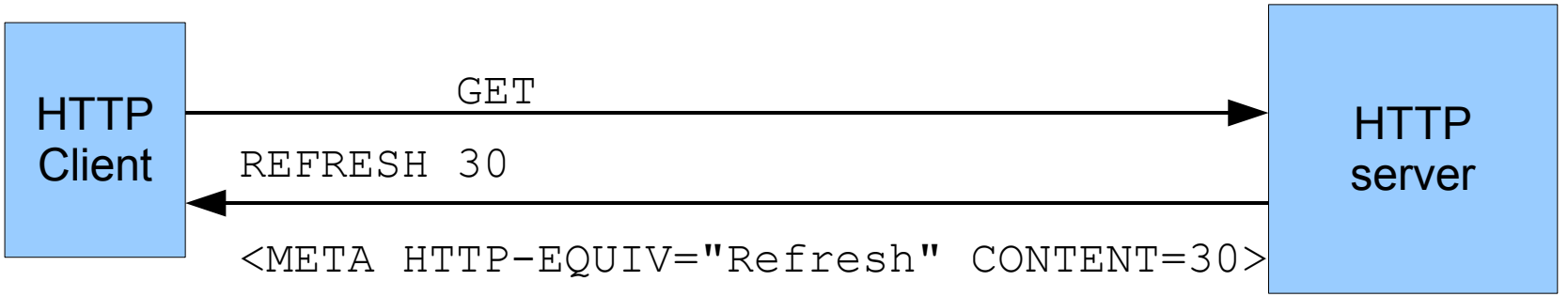
- Latency
  - Seconds, not minutes
- Scalability
  - Millions of clients
- Battery Management
  - Radio vs. Application
- Firewalls
  - Outbound only!
- Routing
- Security
- Interoperability
- Version management
- Video performance
- Multicast scalability
- ...

#1: Polling



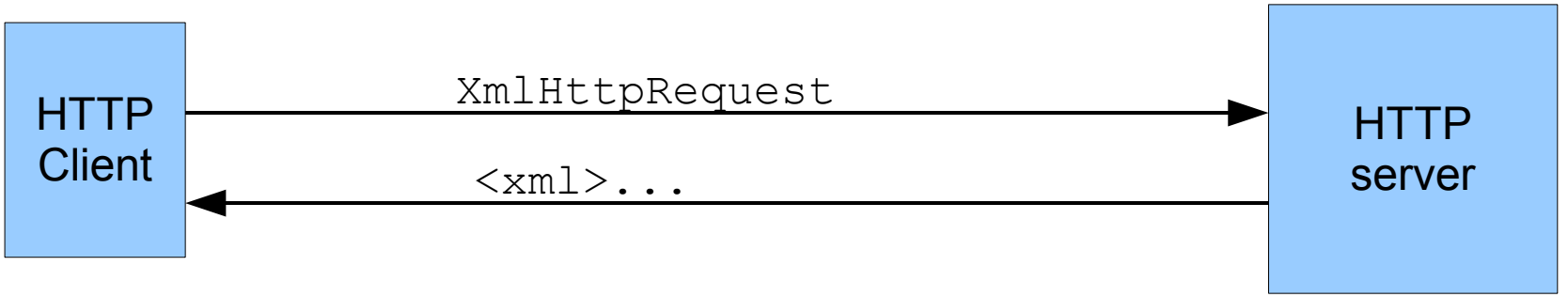
- Check once every *n* seconds
  - Average latency is *n/2*
- Optimizations
  - Vary frequency of polling
  - Compressed response

#1 Polling: HTTP Refresh



- Suitable for thin clients (browsers)
  - Server can control frequency of polling
- Challenges
  - Not supported uniformly
  - Not clear what the browser should do with updates

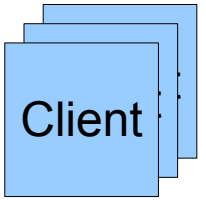
#1 Polling Variant: AJAX

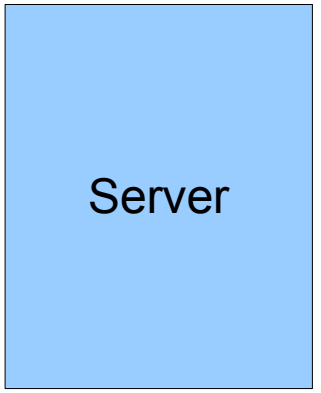


- JavaScript code “polls”
  - Server response arrives asynchronously
  - JavaScript code interprets updates
- Not limited to polling
  - AJAX can do a lot more than polling

#2 Client Listens

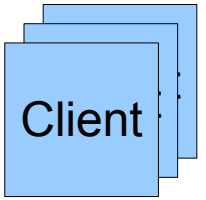
- Registry at Server
  - Client gets dynamic ID
  - Client gets ServerSocket
  - Client informs server
- Server contacts client
  - UDP
  - GSM (SMS)

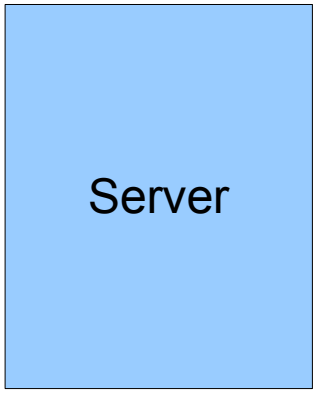




#2 Variant: Poke-n-pull

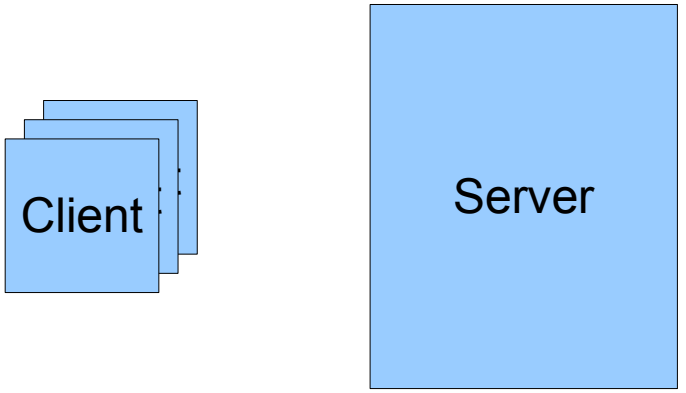
- UDP/SMS inadequate
  - Payload too bulky
  - Update calls for session
- Server “pokes” client
  - By UDP or SMS
  - Client initiates TCP
- Examples
  - WAP Push





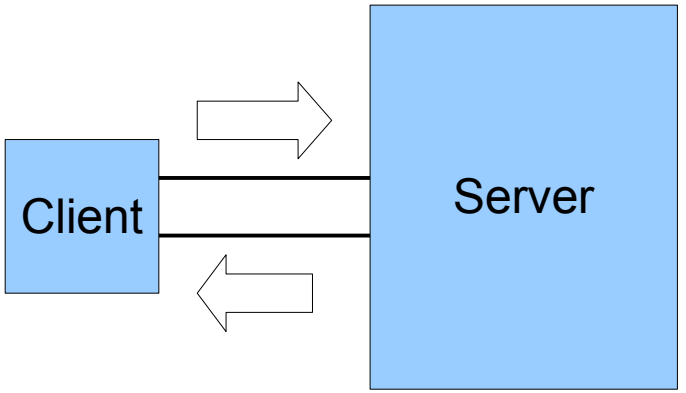
#3 Client stays connected

- Client connects to server
  - TCP, HTTP
- Server keeps conn idle
  - May reconnect repeatedly
  - But connected 24x7
- On push/notifications
  - Server sends data
  - Client processes event



#3: HTTP idiosyncracies

- Why HTTP?
  - Supported on mobiles
  - More robust
  - Gets through firewalls
- HTTP Limitations
  - Request/Response
  - 2 conn/server limit
- New names
  - Comet, reverse-AJAX



#1: Polling: Scalability

- Load on server is O(n*f)
  - *n* Clients
  - *f* Polls/hour
- Not much we can do to help
  - Optimizations (variable *f*, compress data)
  - Distribute over hundreds of blade servers
  - Load balancer respects affinity
  - Move frequent answers closer to the edge

#2 Client Listens: Scalability

- Most scalable of the three
  - Limited only by network topology
  - Direct peer-to-peer connections possible
- However, reality forces spoke-n-hub topology!
  - Hold-n-forward messaging applications
  - Firewalls
  - Limited capability devices
- Hybrid of #2 and #3

#3 Stay Connected: Scalability

- *O(n)* idle connections
  - Workload may be much smaller!
  - Limit on file descriptors/handles
  - Scalability of TCP/IP stack
- Solution
  - Tune transport protocol (TCP)
  - Use async I/O to maximize clients/threads
  - Dynamic load balancing of clients/blade

Recap

- Push continues to be reinvented
  - Many implementations
- Three paradigms
  - Client polls server
  - Client listens on a port, peers and servers talk
  - Client listens to a server, which relays peers
- Scalability
  - Hub-n-spoke topology requires scalable hub