Mobile Computing
#MC03 Data Synchronization

CS60002: Distributed Systems
Winter 2006-2007
What kind of sync?

• “Sync” is an overloaded phrase
  - Not semaphores `synchronized () { ... }`
  - Not shared memory and not cache coherence
  - Not instruction pipelining
  - SyncML, ActiveSync, cvs merge, database replication, ...

• Data synchronization
  - Two copies of data: on mobile and on server
  - Need to keep them in sync
Contact Entry

Name: Ravi Patel
Home: +91-33-2424-5555
Mobile: +91-9999912345
Email: Ravi@xyz.com

Contact Entry

Name: Ravi Patel
Home: +91-33-2424-5555
Work: +91-33-2424-6666
Mobile: +91-9999912345

Data Sync

Wireless Network
### What to sync?

<table>
<thead>
<tr>
<th>Structured</th>
<th>Unstructured</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contacts, Calendar, Tasks, App databases, Config data</td>
<td>Radio Code, OS and VM, Application Code</td>
</tr>
<tr>
<td>SMS, Call Logs, EMail</td>
<td>MP3s, Ringtones, Photos</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Create, Update &amp; Delete</th>
<th>Create &amp; Delete only</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Structured</td>
</tr>
<tr>
<td></td>
<td>Unstructured</td>
</tr>
</tbody>
</table>
Sync Complexity

• Communication Complexity
  – Alice and Bob wonder if they have the same string
  – Can they decide without communicating all n bits?
  – No!

• Assumptions help us optimize
  – Assumption#1: Change log
    • What if Alice and Bob knew what has changed since they had the same string?
  – Assumption#2: Blind faith in digital hashes
    • May miss differences
Types of sync

• File sync
  - Bit buckets. No semantics.
  - Master/Slave only.
  - No conflict resolution

• Application sync
  - Application events
    • e.g., “Meeting postponed”, “task declined”
  - Conflict resolution
  - Bi-directional sync

<table>
<thead>
<tr>
<th>Change log</th>
<th>File</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>cvs merge</td>
<td>cvs</td>
<td>(None)</td>
</tr>
<tr>
<td>SyncML</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HotSync</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ActiveSync</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OTADM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hash-based</td>
<td>r sync</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sync using Change Logs

• Universal applicability
  – Used in File Sync (e.g., cvs merge)
  ✔ Used in Application Sync (e.g., Palm HotSync)

• Both Mobile and Server maintain change logs
  – Log may detail actual create/update/delete entries
  – Or just a timestamped list of modified items
  – Or just a “changed-since-last-sync” marker

• Two types of sync operations
  – Fast Sync (always sync to the same server)
  – Slow Sync (sync to multiple servers)
Fast Sync

- What has changed since we last met?
  - Same item+field changed in both copies => Conflict
  - Jointly decide how the other party gets the edit

- Optimizations
  - Mobile is resource constrained. Server is not.
  - Mobile sends all changed items to server.
  - Server collates and sends edits back to Mobile.
Slow Sync

• Works when Fast Sync does not
  – Mobile is synced to >1 Servers (no “changed” flags)
  – Change log overflowed, or is not trustworthy
  – It has been too long since the last Slow Sync

• Simple, but inefficient
  – Mobile sends all records to Server
  – Server collates, and sends edits back to the client
Optimizations for Change Log Sync

- Use change logs, not just “changed” flags
  - Maintain complete log of changes
  - Log overflow => Next sync is a Slow Sync
- Trickle Sync (aka Replication)
  - Soft-real-time bidirectional stream of change events
- Fall back to Slow Sync
  - Periodically when connected over WiFi or USB
- Resource constraints on Mobiles
  - Battery, RAM, Flash
PDA/Mobile Sync Products

- HotSync
  - Palm OS
- IntelliSync
  - Acquired by Nokia
  - “Anywhere” Server
- ActiveSync
  - Microsoft Mobile
  - Renamed(?) in Vista

- SyncML
  - OMA standard
  - Sync two DOM's
    - PIM, email, code, ...
  - Widely supported
- BlackBerry
  - Proprietary
Hash-based Sync

• No change log required
  – Will sync two snapshots without any history
  – Used for “merge”, and not for bi-directional sync
  – Can be wrong

• Interesting problems
  – Substring identification
    • Cut-n-paste editing
    • “Replace all” edits
  – Lossless compression and other encodings
rsync

- File synchronization problem
  - Bob wishes to have Alice's copy
  - Should Alice just send the whole file to Bob?
  - No. rsync can help
    - Andrew Tridgell, Paul Mackerras, “The rsync algorithm”,
rsync

- Bob
  - Partitions string into fixed blocks of size S
  - Sends weak rolling hash and strong hash for each
- Alice
  - Trusts hashes to locate those blocks
  - Sends stream of “insert block” & “insert data”
- Bob
  - Follows commands to construct Alice's copy
Code/Firmware update

V 3.1
V 3.2
V 3.3
V 3.4

● A special case of sync (or data compression)
  – Mobile to Server: “I have V 3.3 – x + y”
  – Server to Mobile: “Apply these deltas …”

● Constraints
  – Server: Scale to support millions of Mobiles
  – Network: Bandwidth
  – Mobile: Battery, RAM, Flash
Three Problems in Code Sync

1) Small patches
   - Compute small delta from V3.3 to V3.4
   - Plenty of resources, since patches will be reused

2) Data compression
   - Plenty of resources for compression
   - Limited resources for decompression

3) Application of patches
   - Limited “disk” space
   - Rollback on failure
Prefix Sort

“ACAIA”

0: ACAIA
1: CAIA
2: AIA
3: IA
4: A

0: ACAIA
1: CAIA
2: AIA
3: IA
4: A

• Several linear time algorithms
  - e.g., Juha Kärkkäinen and Peter Sanders, “Simple Linear Work Suffix Array Construction”, ICALP 2003

• Can be used to find substring in $O(m+\log n)$
Problem

- Generate small patch from V3.3 to V3.4 binaries
- “Replace all” changes to data & branch addresses

Solution

- Allow 50% mismatches in 8-byte segments
  - Control file (add/insert)
  - Difference file (fix small mismatches)
  - New content file
Recap

• File and Application Sync
• Sync using Change log
  – Fast vs. Slow
• Hash-based Sync
  – Rsync
• Code Sync
  – Prefix sort
  – bsdiff