Network-Aware Application Adaptation for Mobile Hosts

Benjamin Atkin and Kenneth P. Birman
Department of Computer Science
Cornell University
Ithaca, NY
Network-aware adaptation

- Adapting to variable network behaviour
  - Availability of bandwidth
  - Amount and types of traffic
- Structure application communication
  - Rapid adjustment to changes
  - Efficient use of bandwidth
- Principal example: wireless networks
Wireless network characteristics

- Noisy medium leads to high error rates
  - 802.11b is "reliable", but throughput drops
- Mobile hosts experience rapid changes
  - Difficult to predict future bandwidth
System setup

Client

wireless LAN

Server

wired LAN
Application adaptation

- Applications using network must adapt
  - Interactive tasks need good response time
  - e.g. Video, web browsing
- Multiple forms of network adaptation
  - Monitor network characteristics
  - Control bandwidth allocation
  - Poll status of ongoing operations
  - Prioritise concurrent data transmissions
Related work

- Network-level adaptation
  - wireless TCP enhancements: ELN [BPSK97], I-TCP [BB97], M-TCP [BS97]
  - congestion control/bandwidth allocation: Congestion Manager [ABCSB00], SCTP [RFC2960]
- Web browsing and video streaming
  - Transcoding through proxies: TACC [FGBA96], ...
  - Quality adaptation for streaming video [RHE99, ...]
- Application support
  - Odyssey [BN99]
  - Rover [ATK97]
  - HATS [LWZ02]
- Distributed file systems
  - Coda [KS92], Little Work [HH95]
  - Ficus [PGH+98,KP97]
  - Low-Bandwidth File System [MCM01]
Modal adaptation

- Small number of modes
  - each corresponds to data quality/fidelity
  - bandwidth determines mode
- Can waste bandwidth
- Changing modes could be expensive
- Not all applications have natural modes
Modeless adaptation

- Variable bandwidth or variable traffic
- Multiple classes of communication
  - e.g. HTML, images
  - not equal importance
- Fine-grained, bursty
  - make decisions on a per-message basis
Adaptation by autoconfiguration

- Modeless adaptation reconfigures communication automatically
  - communication "policy" changes with bandwidth
  - analogous to discovering modes
  - no need to assign thresholds
Adaptive applications

- Well-suited for RPC client-server applications
  - Distributed file system
  - E-mail client
  - Publish-subscribe
  - Web browsing
  - Web services with "differentiated service" to clients
Applying modeless adaptation

- Two levels of adaptation investigated:
  - ATP: adaptive transport protocol [Infocom 2003]
  - MFS: adaptive distributed file system
Adaptive Transport Protocol (ATP)

- Message-oriented
  - Reliable delivery
- Bandwidth notifications via upcalls
- Priorities to determine send order
- Allows speculative communication
  - High priority preempts low priority
  - Inessential activity can run "in background"
  - e.g. Prefetching files
ATP implementation

application

network scheduler

bandwidth estimator

message queue

endpoint

[Graph showing network bandwidth over time]
ATP implementation

- application
- message queue
- network scheduler
- bandwidth estimator
- endpoint
ATP adaptation

- Supports multiple adaptation styles
  - Modal using bandwidth notification
  - Modeless using send timeouts
  - Modeless using priorities
Modal versus modeless adaptation

- Compared using a video-like workload
- Transfer data at 4 levels of quality
  - level 1 lowest, level 4 highest
- Modal: 4 modes, bandwidth estimates
- Modeless: differential encoding with priorities
  - transfer level 2 messages iff level-1 all delivered, etc
Modal versus modeless adaptation: example

Modal adaptation

Modeless adaptation

higher utilisation
Modal versus modeless adaptation: explanation

Modeless adaptation

Modal adaptation
Mobile File System (MFS)

- AFS-style client-server design
  - Whole-file client-side caching
  - Stateful file server
  - Writeback-on-close semantics
- Modeless adaptation via ATP
- Filesystem-specific adaptation mechanisms to improve performance
MFS cache manager

Applications

open a.txt

incoming traffic

demand fetch

prefetch

outgoing traffic

synchronous writeback

asynchronous writeback

background processing

cache consistency

access monitoring

Adaptive RPC library

Server

reply

fetch RPC

Network-aware Application Adaptation for Mobile Hosts / Benjamin Atkin
Coda-style modal adaptation

- Strongly-, weakly-connected or disconnected
- Weakly-connected mode
  - Log writebacks at low bandwidth
  - Periodic log flushes reduce traffic
- "Unpredictable" changes in semantics
MFS adaptation to bandwidth variation

- Each RPC type has a priority
  - "Background operations" => low priority
  - "Interactive operations" => high priority
- ATP favours high-priority RPCs
- Background asynchronous writeback at all bandwidth levels

| VALIDATE (high) | FETCH DATA | METADATA | STORE DATA | PREFETCH (low) |
Validate workload

Validate/Compile

Validate/Write

MFS priorities improve completion time of small RPCs

foreground: small RPCs
background: large RPCs, CPU-bound

foreground: small RPCs
background: large RPCs, I/O-bound
Read workload

Read/Compile

Read/Write

MFS priorities improve completion time of large RPCs
Background prefetching

- Extension from local file systems
  - Use surplus bandwidth speculatively
- Relies on prefetching hints
  - e.g. Application dependencies
- Automatically-generated file groups
  - Accessing a member triggers prefetching
  - Implemented as a special file type
Examples of prefetching

Multigrep

Simultaneous writeback

Prefetching can be highly beneficial, rarely a liability
### Prefetching statistics

<table>
<thead>
<tr>
<th>test</th>
<th>elapsed time</th>
<th>traffic</th>
<th>good prefetches</th>
<th>bad prefetches</th>
</tr>
</thead>
<tbody>
<tr>
<td>multigrep</td>
<td>28%</td>
<td>1.9 MB</td>
<td>1.9 MB</td>
<td>0.0 MB</td>
</tr>
<tr>
<td>pause</td>
<td>40%</td>
<td>8.0 MB</td>
<td>5.9 MB</td>
<td>0.0 MB</td>
</tr>
<tr>
<td>sim. demand</td>
<td>73%</td>
<td>8.0 MB</td>
<td>3.9 MB</td>
<td>0.1 MB</td>
</tr>
<tr>
<td>sim. writeback</td>
<td>69%</td>
<td>8.0 MB</td>
<td>3.8 MB</td>
<td>0.1 MB</td>
</tr>
</tbody>
</table>

Large think time increases the possibilities for prefetch-computation overlap
Prefetching statistics

<table>
<thead>
<tr>
<th>test</th>
<th>elapsed time</th>
<th>traffic</th>
<th>good prefetches</th>
<th>bad prefetches</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward order</td>
<td>88%</td>
<td>8.0 MB</td>
<td>7.8 MB</td>
<td>0.0 MB</td>
</tr>
<tr>
<td>reverse order</td>
<td>76%</td>
<td>8.0 MB</td>
<td>3.6 MB</td>
<td>0.0 MB</td>
</tr>
<tr>
<td>bad groups</td>
<td>132%</td>
<td>4.0 MB</td>
<td>0.0 MB</td>
<td>18.1 MB</td>
</tr>
</tbody>
</table>

"Fast-linear-scan" can reduce benefit

Poor prefetching hints can decrease performance (rare case)
MFS: future work

- Reconciling asynchronous writeback with strong cache consistency
- Consistent access to stale files
- Automatic derivation of caching policies from file usage patterns
- Performance under actual use
Summary

- Modeless adaptation provided by ATP
  - Improves bandwidth utilisation
  - Allows fine-grained adaptation at RPC level
- MFS: modeless adaptation in a file system
  - Priorities for RPC types using ATP
  - Speculative communication mechanisms to improving performance (e.g. prefetching)