CS5984
Mobile Computing

Dr. Ayman Abdel-Hamid
Computer Science Department
Virginia Tech
Adaptive Mobile Applications in Odyssey

Outline

• Experiences with adaptive mobile applications in Odyssey

Application-aware Adaptation

• Odyssey: Platform for mobile data access

• Resources available to mobile clients vary widely and unpredictably
  ➢ Client has to adapt
    ✓ reduce demand for a scarce resource by substituting a more plentiful one
    ✓ lossless compression to data (trade computation for bandwidth)
    ✓ Caching of data (trade disk space for bandwidth and computation)
    ✓ Reduce quality of data it is accessing, hence reducing resource consumption

• Application-aware adaptation: collaboration between applications and operating system to provide adaptation

Data Fidelity

• Odyssey’s approach to adaptation
  ➢ Trading data quality for resource consumption
  ➢ When bandwidth to video player drops, switch to a video stream with fewer colors and coarser resolution (rather than suffer dropped frames)
  ➢ When bandwidth to a Web browser drops, can fetch images that have been aggressively compressed rather than wait for full quality versions

• Reference copy: most complete, detailed, current version of an item ➔ attempt to always present reference copy to mobile client

• Data fidelity: degree to which a presented item matches the reference copy
  ➢ A type-specific notion
  ➢ Video may be degraded by dropping frames, or reducing quality of individual frames
Adaptation Model

- Application-aware adaptation
  - The OS responsible for
    - Monitoring resource availability
    - Notifying applications of relevant changes to resources
    - Enforcing resource allocation decisions
  - Application responsible for deciding how to exploit available resources
- Laissez-faire adaptation: application fully responsible
  - No support for concurrent applications (compete for same scarce resources)
- Application-transparent adaptation: system entirely responsible
  - Does not support diversity (different applications can react differently to same data in same circumstances)

Client Architecture

- Odyssey objects available to applications as part of file name space
- Interceptor forwards file system requests on Odyssey objects to viceroy
- Viceroy responsible for all type-independent functionality on the client
  - Monitor availability of resources
- Warden: type aware code components (1 per type)
  - Provide a menu of fidelities from which application can pick
  - Semantic access methods

Programming Interface 1/2

- Two calls added to standard system API
- Resource request
  - Inform Odyssey of changes to resource availability (video player wishes to know when bandwidth drops below a certain level)
  - Lower/upper bound: window of tolerance
- Type-specific operation
  - Used by application to change fidelity at which data is accessed
  - Video application reduces frame rate

Programming Interface 2/2

- Application chooses a fidelity level, issues a resource request (specify a tolerance window, and a function to call when resource is outside the tolerance window)
- Resource requests forwarded to viceroy
- Viceroy updates estimates of resource availability
- If resource outside tolerance window, affected application notified via an apcall (treated similar to signals but with exactly-once semantics)
- Application responds by changing fidelity of data
- Fidelity changes carried out by wardens (tsop)
- Tsop mechanism can provide type-specific access methods (read a movie as frames of video rather than bytes)
Managing Bandwidth

• Bandwidth considered most volatile resource for mobile clients

• Bandwidth estimation algorithm
  - Each exchange between client and server is timed and logged
  - Prediction of near-term total bandwidth available to machine

• Total bandwidth divided amongst applications competing for it
  - Most of bandwidth is divided based on recent use
  - A small portion is reserved and divided fairly

Example Applications: Video Player (Xanim) 1/2

• Source code publicly available
• Code split into client, warden, and server
• Video server stores each movie as a number of pre-computed versions called tracks
• Prototype Implementation
  - Movie consists of 3 tracks (JPEG-compressed color at quality 99, JPEG-compressed color at quality 50, black-and-white)
  - Each frame encoded at 10 frames/second

Example Applications: Web browser

• Source code not available then
• Use of Netscape Proxy facility (route all requests through a designated process)
• Cellophane is the adaptive application
• Distillation server fetches HTML pages and images and can degrade images on the fly using JPEG compression (4 levels)
• Load best quality image possible within twice the expected time to load reference image at 10 Mbps
Example Applications: Speech Recognizer (Janus)

- Expensive process in terms of CPU cycles and virtual memory
- Speech recognition: vector quantization + remaining tasks
- Remote recognition: pass in full to remote
- Local recognition: pass in full to local
- Hybrid recognition: pass to local for vector quantization, then pass small result to remote
- Decide which is more suitable based on available network bandwidth

![Diagram of Janus with different recognition processes]