

Power-Aware Multimedia Streaming in Heterogeneous Multi-User Environments*

Radu Cornea, Shivajit Mohapatra, Nikil Dutt,
Alex Nicolau, Nalini Venkatasubramanian

Information and Computer Science
University of California, Irvine

*This work was partially supported by NSF award ACI-0204028

Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- System Architecture
- Video Quality Perception
- Low Level Optimizations
- System Level Optimizations
- Experiments, Results
- Related Work
- Conclusions and Future Work

Introduction

- Technological advances
 - Processor
 - Wireless networking
- New class of applications for mobile devices
 - Multimedia streaming
- Portable devices have limited resources
 - Processing power, memory
 - Smaller display size
 - Internal storage
 - Limited battery life



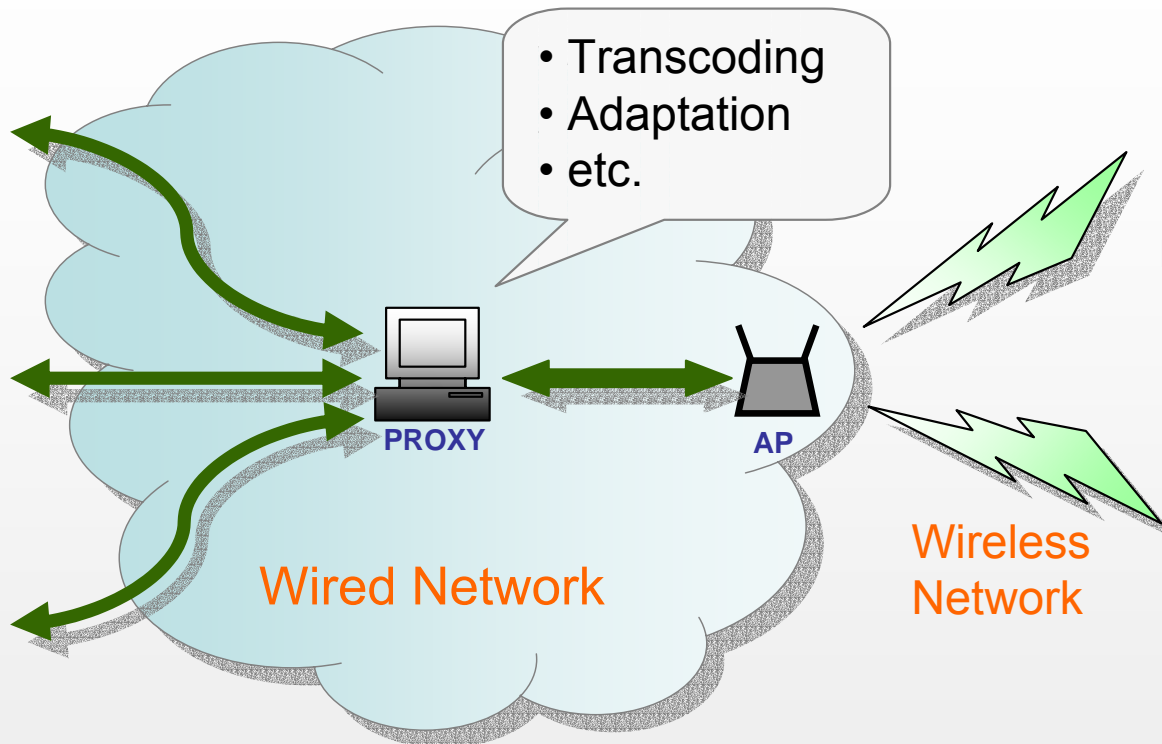
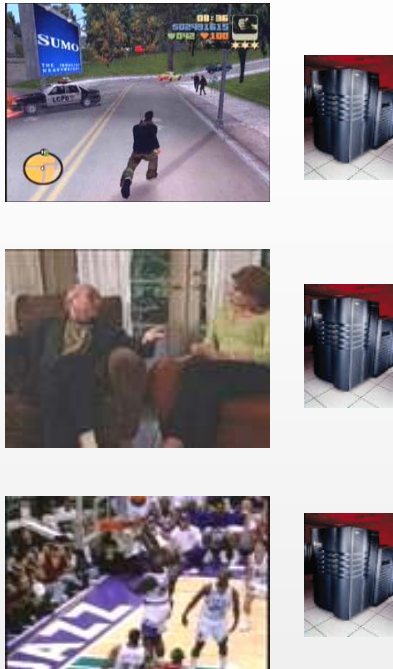
▶ Streaming framework needs to be power-aware

Outline

- Introduction
- **Multimedia Streaming: Problems, Solutions**
- Our Approach
- System Architecture
- Video Quality Perception
- Low Level Optimizations
- System Level Optimizations
- Experiments, Results
- Related Work
- Conclusions and Future Work

The Streaming Process

MEDIA SERVERS



CLIENTS



Handheld PC



PDA

- Proxy-based streaming
- Multiple stream sources
- Heterogeneous network of client devices

Problem and Solutions

- Multimedia vs general purpose applications

- High QoS and processing requirements

- Heterogeneous, distributed environment

- Pose challenges for the system designer

- Solutions – at different abstraction levels

- Architecture: cache, memory optimizations

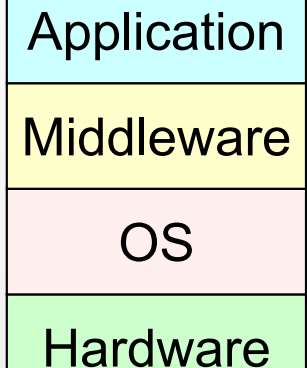
- Dynamic voltage scaling (DVS)

- Dynamic power management (DPM)

- System components: disks, network interfaces

- Compiler techniques, applications/middleware adaptations

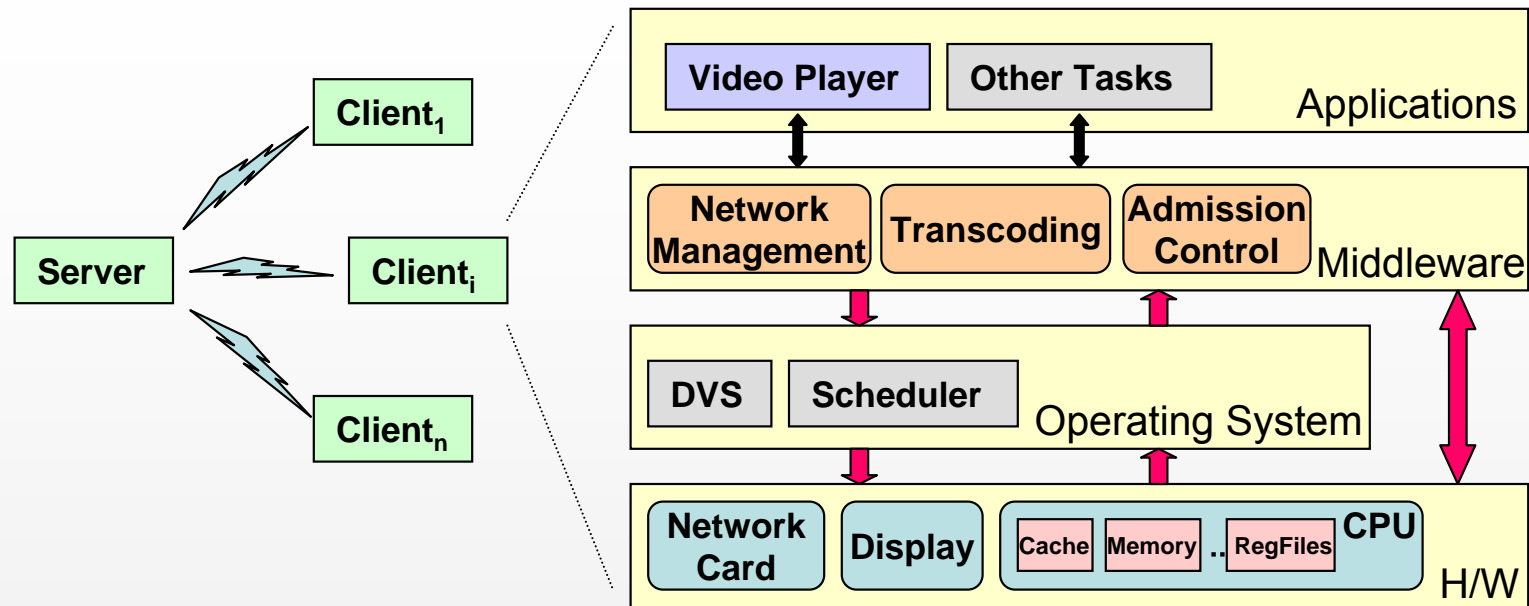
- Disconnect between techniques at different levels



Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- **Our Approach**
- System Architecture
- Video Quality Perception
- Low Level Optimizations
- System Level Optimizations
- Experiments, Results
- Related Work
- Conclusions and Future Work

Our View: Layers & Interactions



Abstraction Layers

- Architecture: local view
- Middleware: global view

- Goal: develop and integrate hardware level with middleware level techniques
=> Cumulative power gains

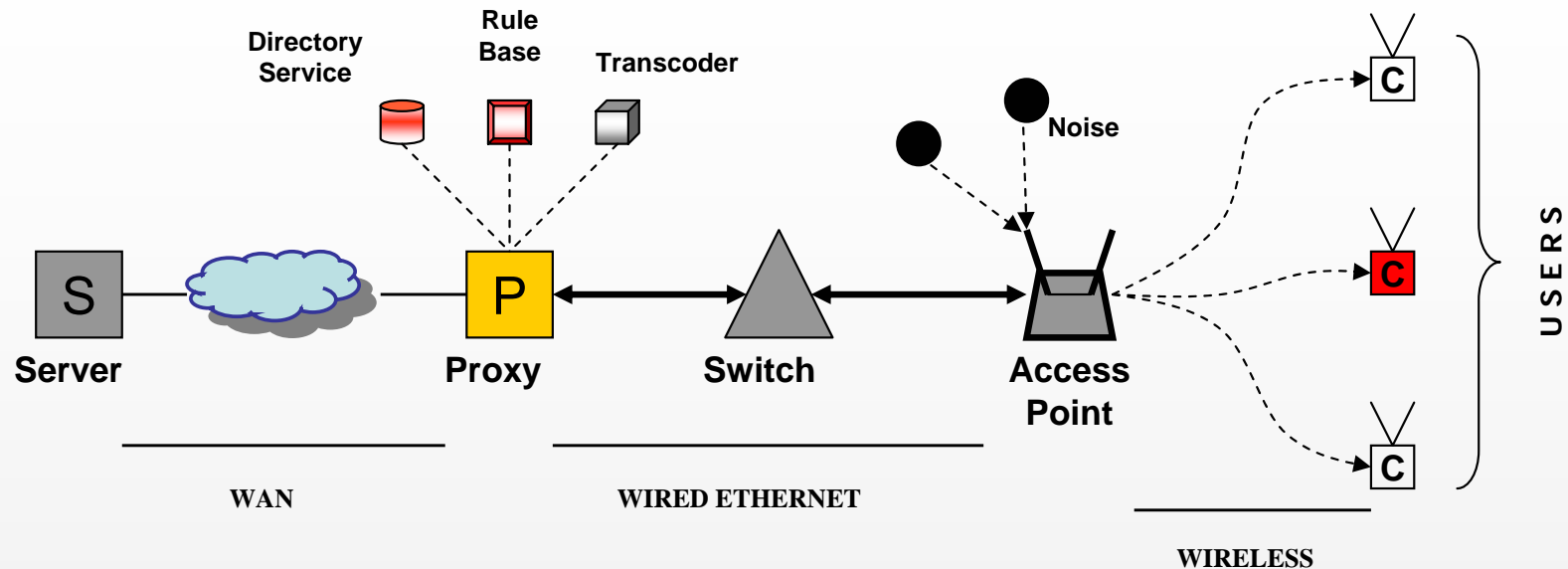
Approach

- Phases in our approach:
 - Identify low level architectural knobs and compilation techniques
 - Determine optimized operating points for these knobs
 - Study DVS tradeoffs
 - Evaluate power gains of wireless network
 - Estimate effect of multiple users - resource limitations
 - Integrate framework and admission control
- Study video quality and power tradeoffs
- Improve user experience, system performance

Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- **System Architecture**
- Video Quality Perception
- Low Level Optimizations
- System Level Optimizations
- Experiments, Results
- Related Work
- Conclusions and Future Work

System Architecture



- Server - client model, proxy in between
- Noise: models the other users in the system
- Middleware level controls the system
 - Executes on proxy, client devices

Middleware Layer

- Middleware functions:
 - On the client device
 - Sends device information to the proxy (e.g. energy)
 - Relates video stream/network parameters to lower abstraction layers
 - On the proxy node
 - Power-aware admission control
 - Real-time transcoding
 - Controls network transmission

Proxy Node

- Proxy - components
 - Transcoder: adapt stream to device specific capabilities
 - Rule base: static device specific info (architecture, OS, etc.)
 - Directory service: dynamic global state information
 - Mobility info
 - Noise level
 - etc.
- Middleware exploits info from rule base and directory to perform its functions

Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- System Architecture
- **Video Quality Perception**
- Low Level Optimizations
- System Level Optimizations
- Experiments, Results
- Related Work
- Conclusions and Future Work

Goal: User Experience

- Ultimate goal
 - Provide an improved end user experience
 - Maintain an acceptable utility factor for system
- What is “acceptable utility factor” ?
 - User can stream the video to completion, at the highest possible quality
- Important !
 - Video quality: human perception is subjective
- Observation: user perception of quality highly influenced by environment and device
 - On desktop/laptop much easier to differentiate between close levels than on handheld device



Video Quality Levels

- Hard to programmatically identify video parameters that produce a user perceptible change in quality
 - Approach: select video quality levels such that there is a visible degradation between levels
 - 8 quality levels (Q1 - Q8) chosen
 - Parameters varying: frame size, bitrate, frame rate
 - Profile power for each quality
 - Optimize system for each quality
- } Select best tradeoff

Example



Q1



Q2



Q3

Outline

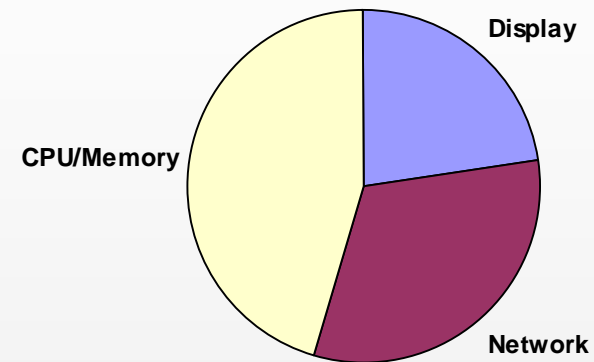
- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- System Architecture
- Video Quality Perception
- **Low Level Optimizations**
- System Level Optimizations
- Experiments, Results
- Related Work
- Conclusions and Future Work

Multimedia Streaming: Optimizations

- **Complex distributed environment**
 - Requires coordination at different levels of abstraction
- **Method applied:**
 - Start with a single client - proxy - server model
 - Perform possible optimizations: hardware, network
 - Move up and study system level adaptations
- **Levels of optimizations:**
 - Coarse grain adaptation: select best video quality to be streamed (middleware)
 - Fine grain tuning: adapt architecture to chosen video quality level

Hardware Level Adaptation

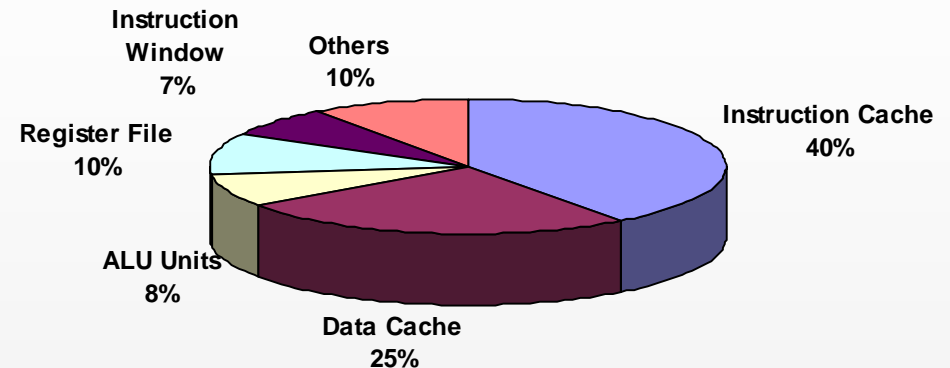
- Question: How is power consumption distributed?
- Profiling => Power breakdown:
 - Display ~ 1W
 - Network ~ 1.4W
 - CPU/Memory ~ 1-3W
 - Challenging, data dependent
 - But good results can be obtained
- Architecture level knobs explored
 - Cache configuration
 - Dynamic Voltage Scaling (DVS)
 - Network traffic control



Cache Reconfiguration

- Caches (instruction and data) account for largest part of CPU energy

Cache { Instruction ~ 40%
Data ~ 25%



- Idea: adapt cache configuration to specific video stream requirements
 - MPEG decoding: poor cache locality, large data sets => increased cache-memory traffic
- Cache power - depends on size and associativity
 - Determine best configuration for each quality level

Integrated DVS

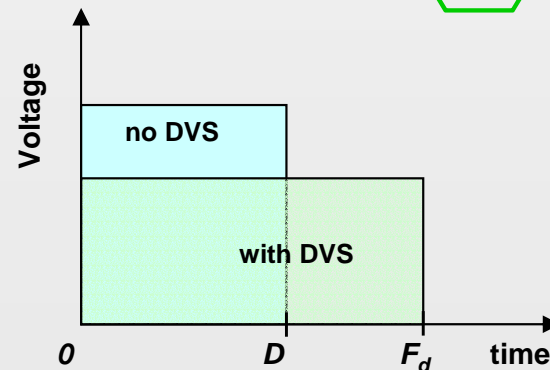
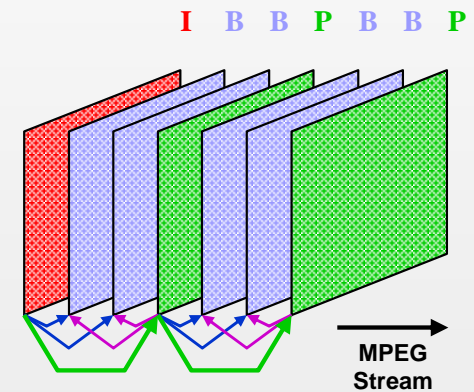
- DVS idea: trade off processor speed for power
- MPEG frame decoding – good candidate
 - Frame decoding takes less than the frame delay
 - Decoding time – depends on frame type: I, P, B

- Assumption:

- Buffered decoding
 - Tolerates small deadline misses

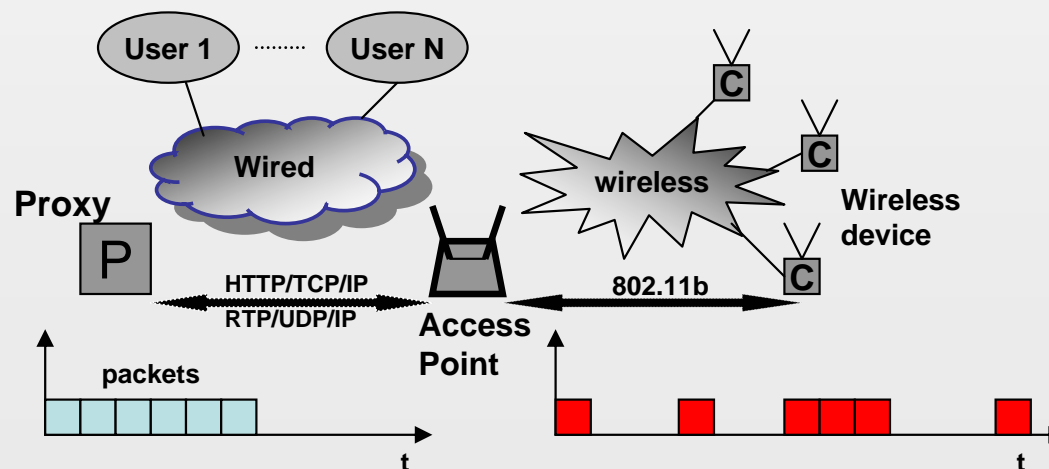
- Slack time: $\theta = F_d - D$

F_d = frame delay
 D = decoding time



Network Traffic Regulation

- Proxy-based traffic shaping
 - Adapt to network conditions and wireless transmission attributes (bandwidth, buffering)
- Optimized burst transmission
 - Allows for long sleep intervals for the network card
 - Control info added - when device should wake up
 - Large burst lengths can result in loss of packets



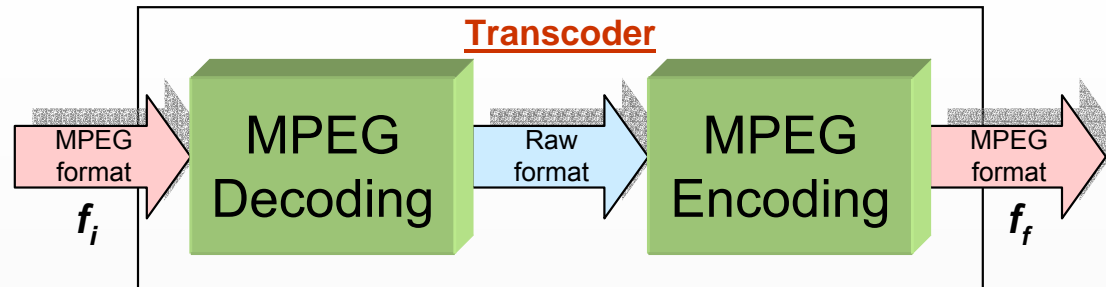
Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- System Architecture
- Video Quality Perception
- Low Level Optimizations
- **System Level Optimizations**
- Experiments, Results
- Related Work
- Conclusions and Future Work

System Level Optimizations

- Heterogeneous, multi user environment
 - Large variation in playing capabilities
 - Processing power, display size, battery life, etc.
- New problems
 - Network congestion (limited bandwidth available)
 - Limited resources (e.g. proxy node CPU load)
- Multiple users effects
 - Proxy : transcoder
 - Access point : network bandwidth
- System level admission control

Transcoding



- Two steps:

- MPEG decoding => raw format (YUV, YV12, PPM)
- MPEG encoding into a different format (after some processing - change size, frame rate, bit rate)
 - Most processing requirements

- Load on proxy: $L_{trans}(f_i, f_f) = L_{dec}(f_i) + L_{enc}(f_f)$

- Load requirements: $\sum_n L_{trans}^i \leq 1$

L_{trans}^i - load for transcoding the stream for client i

Network Bandwidth

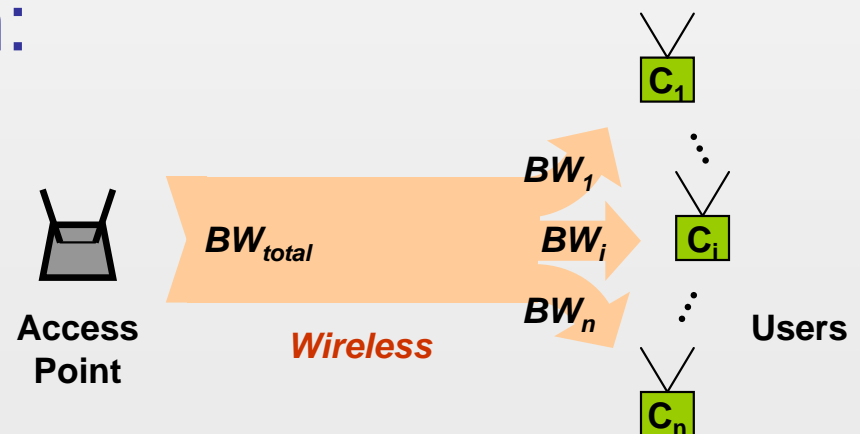
- Wireless network bandwidth is limited
 - Network congestion with many users
- Traffic regulation (extension from single user)
 - Each client transfers burst of P_b bits, every I seconds

Bandwidth is: $BW_i = P_b^i / I^i$, for client C^i

- Avoid network congestion:

$$\sum_n BW_i \leq BW_{total}$$

- Average bandwidth
 - Local buffering



Multi-User Admission Control

- **Middleware continuously monitors the system**
 - **Receives periodic feedback from client**
 - Device status (battery level, load, etc.)
 - **For each client joining performs admission control**
 - **Streams highest level of video quality that meets constraints**
 - Available residual energy, minimum acceptable quality level (user)
- **Heterogeneous environment**
 - **Large diversity of clients**
 - **Device capabilities stored for each client**

Admission Control Info

- Information stored for each client and quality level
 - **Architecture:**
 - Optimized knob configuration (cache parameters, DVS)
 - Average CPU power consumption (profiling)
 - **Network:**
 - Ideal burst length and corresponding network level power consumption
 - **Proxy:**
 - CPU load for converting initial stream to quality Q_i
 - **Access point:**
 - Bandwidth for streaming video at quality Q_i

Admission Control Steps

- Device information:
 - Static: when new client joins
 - Dynamic: precomputed by proxy middleware
- Control called on user join or feedback receive
- Steps:
 - Can accommodate client?
 - If yes, start with minimum quality levels
 - Increase quality level gradually
 - Set new operating point
- Order of client selection is important
 - Heuristic: priority for clients with low video quality

Admission Control Algorithm

```
WHILE (TRUE)
{
  IF ( $R_i$  OR  $F$  received)
  {
    FOR each client
    {
      Determine  $k$ , such that  $Q_k = Q_a$ 
      IF  $((T - (T_{cur} - T_{start})) * P_k \leq E_{res})$  (1)
        Compute  $L_{trans}^i$ ,  $BW_{AP}$  for video quality  $Q_k$ 
      ELSE
        REJECT the request OR (Negotiate video quality)
    }
    IF  $(\sum L_{trans}^i \leq 1 \ \&\& \ \sum BW_{AP}^i \leq BW_{total})$  (2)
    {
      WHILE (at least one change in the client configurations)
      {
        FOR each client  $C_i$  (use heuristic to select)
        {
          Increase quality by 1:  $Q_k \leftarrow Q_k + 1$ 
          IF (Inequalities (1) and updated (2) hold)
            CHANGE quality to  $Q_k$  for client  $C_i$ 
        }
      }
      UPDATE system with new client configurations
    }
    ELSE
      REJECT the request OR (Negotiate video quality)
  }
}
```

For each new event (user join, feedback received)

Try to accommodate every client with the minimum acceptable quality level

If possible, gradually increase quality levels, as long as all constraints are met

Otherwise, reject user (cannot admit into system)

Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- System Architecture
- Video Quality Perception
- Low Level Optimizations
- System Level Optimizations
- **Experiments, Results**
- Related Work
- Conclusions and Future Work

Performance Evaluation

- Steps:

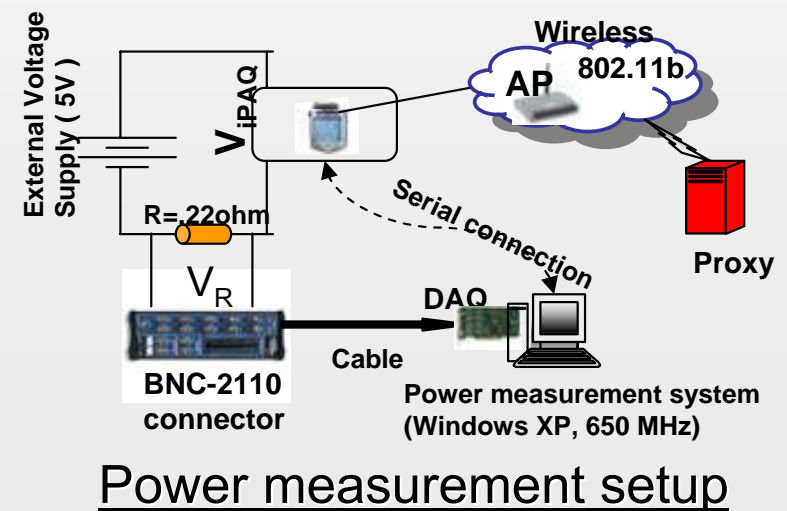
- Select 8 quality levels (Q1 – Q8)
- Optimize architecture for the levels
- Identify best network transfer parameters
- Estimate transcoding loads between formats
- Evaluate admission control algorithm
 - Combination of all above

- Quality levels selected

- Q1 (highest): 320x240, 30fps, 650kbps
- ...
- Q8 (lowest): 160x120, 20fps, 100kbps

Experimental Setup

- Power measurements:
 - IPAQ 3650 + Cisco 350 Aironet wireless card
 - 206Mhz Intel StrongArm, 16MB ROM, 32MB RAM
- Simulation
 - Wattch / SimpleScalar for ARM
- MPEG decoder: Berkeley MPEG tools
- Transcoder: TMPGEnc
- Video clips
 - High action (e.g. GTA)
 - Medium action (sport)
 - Low action (news)



Architectural Optimizations Results

- Profile short (10sec) video clips (Q1 – Q8) for all combination of cache parameters
 - **Size:** 4KB – 64KB
 - **Associativity:** 1 – 32
- **Power savings:** 10-15%
- Obs:
 - **Associativity** - largest power change
 - **Best cache configuration reflects**
 - internal storage requirements for different frame sizes
 - decoding algorithm internal organization

Cache Configuration + DVS

- Combine cache technique with DVS for an increased effect
- Results: up to **60%** energy savings
- Could run CPU at significant lower voltage
=> **Power savings**
- E.g. for action type video clip

Video Quality	Cache Size	Cache Associativity	Clock Frequency	Voltage	Original Energy	Optimized Energy	Savings
Q1	8	8	100	1	1.29	0.76	47.50%
Q2	8	8	100	1	1.09	0.64	47.80%
Q3	8	8	100	1	0.95	0.56	48.00%
Q4	32	2	66	0.9	0.54	0.26	57.60%
Q5	32	2	66	0.9	0.48	0.23	57.80%
Q6	32	2	33	0.9	0.42	0.2	58.00%
Q7	8	8	33	0.9	0.29	0.14	57.30%
Q8	8	8	33	0.9	0.24	0.11	57.50%

Base configuration:

- Frequency 400MHz
- Voltage 1.3V

Network Optimization Results

- Assumptions
 - 10Mbps wired, 8Mbps wireless (effective)
- Varied burst time, noise level, packet size
- Obs:
 - High quality levels have small optimized burst time
- Ideal burst time \Leftrightarrow no frame misses
- E.g. for action type video clip

Quality Level	Burst Length (N=1, secs)	Power Saved (N=1, Watts)	Burst Length (N=3, secs)	Power Saved (N=3, Watts)	Burst Length (N=5, secs)	Power Saved (N=5, Watts)
Q1	2.3	0.925	2	0.89	1.8	0.87
Q2	3.5	1	3.05	0.98	2.76	0.96
Q3	4.6	1.04	4.05	1.02	3.68	1
Q4	4.85	1.05	4.2	1.03	3.85	1.02
Q5	6.8	1.08	6.25	1.07	5.75	1.06
Q6	14.5	1.12	12.5	1.11	11.5	1.11
Q7	17.5	1.13	15	1.12	13.5	1.11
Q8	17	1.12	15.4	1.12	14	1.11

Transcoding Results

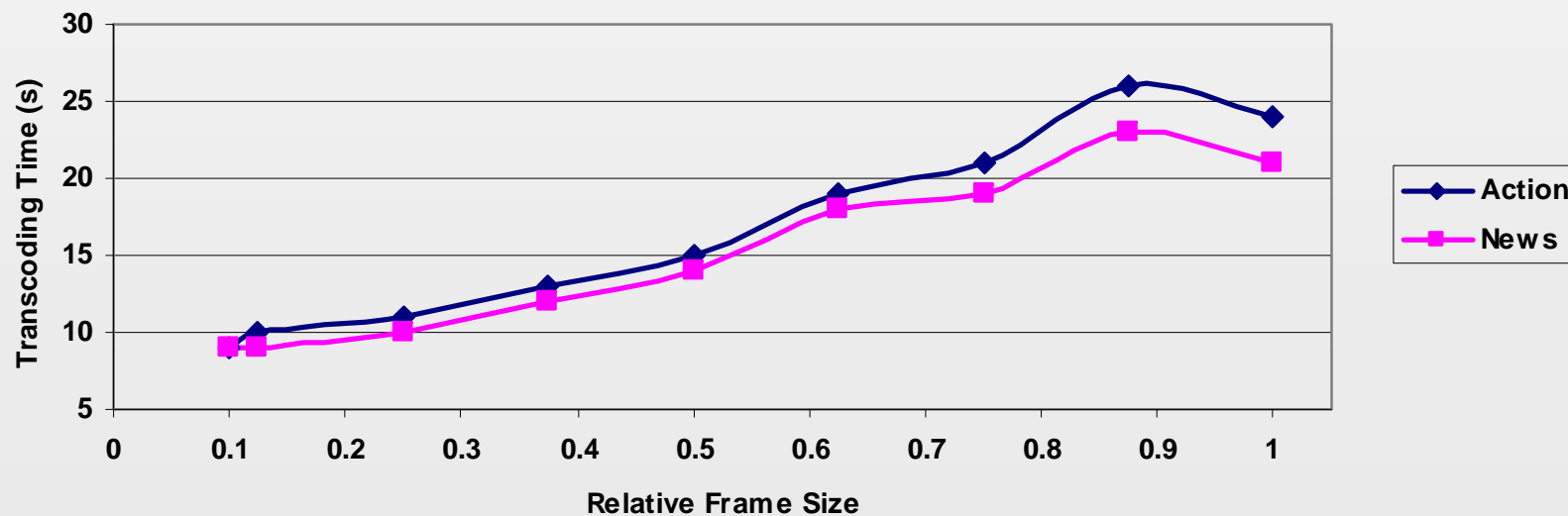
- **TMPGEnc transcoder**

- Hardware details: P4, 2.4Ghz, 512Mb memory

- **Obs:**

- Frame size has largest influence in transcoding time (proxy load)

- e.g. for action and news-type video clips:



Transcoder Load Results

- Other observations

- Bitrate does not affect transcoding significantly
- Nature of video stream affects total transcoding time

- Transcoding time => proxy load

- Let total time be t_{trans} in single user mode (100% CPU available), for a 60 sec video clip
- In multi-user, real-time,

CPU load is $L_{trans} = t_{trans} / 60$

- e.g. for action / news type video clip:

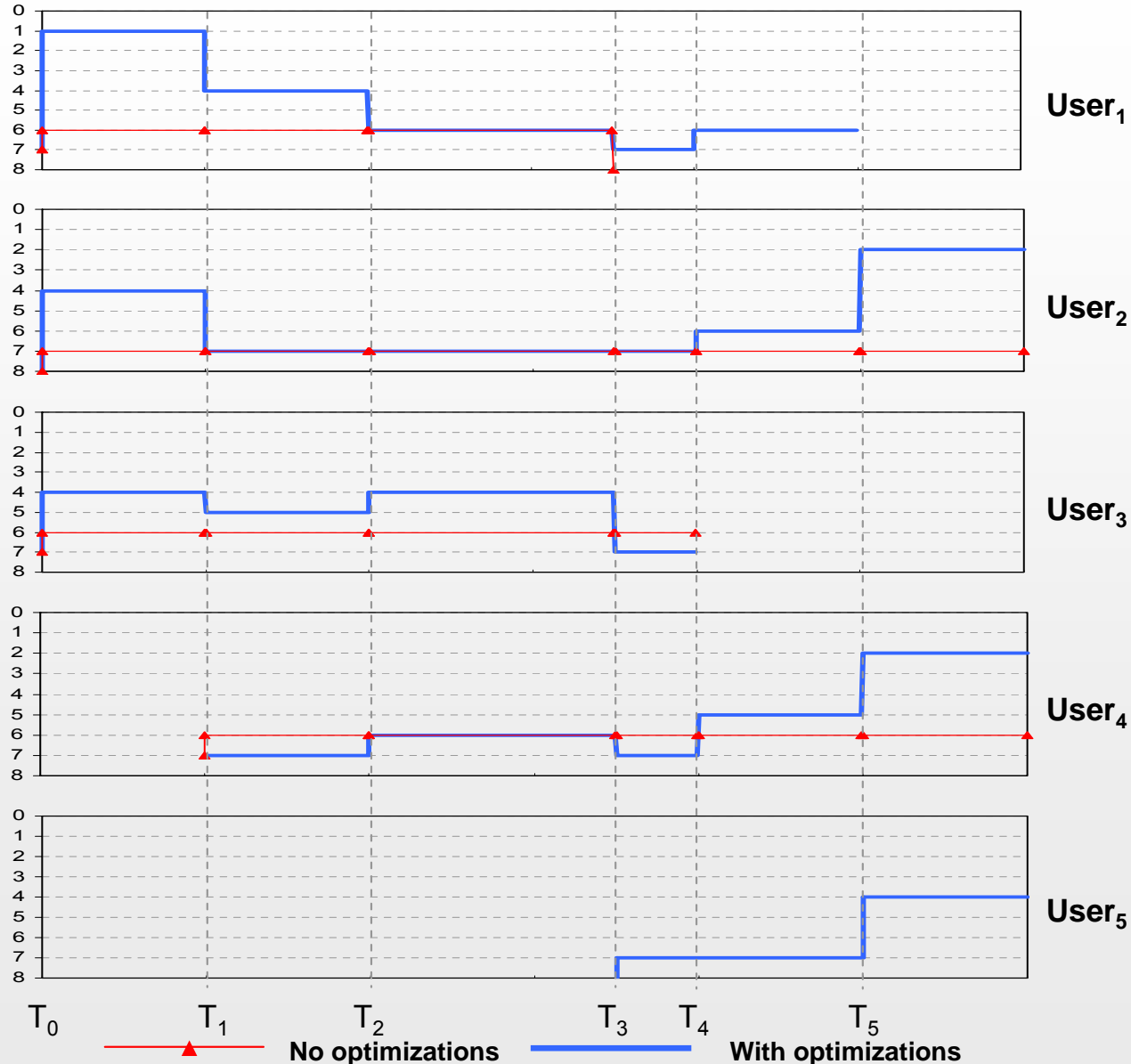
Video	Proxy Load	
	Action	News
Q1	0.4	0.35
Q2	0.4	0.35
Q3	0.4	0.35
Q4	0.3	0.28
Q5	0.3	0.28
Q6	0.3	0.28
Q7	0.2	0.18
Q8	0.2	0.18

Integrated Framework

- Combine optimizations at all levels
- Employ multi-user admission control algorithm
- Dynamic scenario
 - Sensitive to changes in environment:
 - New user joins
 - Feedback in device residual power
 - MPEG streaming ends
 - Low user acceptable qualities (Q7, Q8)
 - Allows for a larger range of operating points
- Initially there are 3 users in the system, later 2 more users join, and 2 users finish streaming

Dynamic Scenario

Video Quality



- T₀: 3 users (1, 2, 3)
- T₁: user 4 joins
 - System readjusts quality levels
- T₂: residual power on user 1 decreases
 - Quality level is decreased for 1
- T₃: user 5 joins
 - All levels go down to accommodate 5
- T₄: user 3 finishes streaming
 - Quality levels rise back
- T₅: user 1 finishes streaming
 - Even higher levels

Scenario Results

- Without optimizations:
 - Only 4 clients could be served by system
 - User 1 could not stream entire video
 - Lower video qualities for the rest of the users
- With optimizations and admission control:
 - System is able to accommodate more users
 - Lower video quality levels for the other users
 - Provides minimum quality required by users
 - Stream quality increases
 - Due to combined optimizations at both levels (hardware, middleware)

Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- System Architecture
- Video Quality Perception
- Low Level Optimizations
- System Level Optimizations
- Experiments, Results
- **Related Work**
- Conclusions and Future Work

Related Work

- Architecture / OS level:
 - Exploit locality in cache
 - Soderquist and Leeser (1997)
 - DVS for MPEG streams
 - Choi et al (2002)
 - Mesarina and Turner (2002)
 - Frame traversal reordering (improve cache hits)
 - Feng and Sechrest (1996)

Related Work (contd)

- **Middleware / Application Level**
 - **Study network power consumption (wireless)**
 - Chandra (2002)
 - Feeney and Nilsson (2001)
 - **Cross-layer adaptation (coarse and fine grained tuning)**
 - Yuan et al - GRACE project (2003)
 - **Proxy based video transformations**
 - Shenoy and Radkov (2003)
 - **Dynamic transcoding techniques**
 - Acharia and Smith (1998)

Outline

- Introduction
- Multimedia Streaming: Problems, Solutions
- Our Approach
- System Architecture
- Video Quality Perception
- Low Level Optimizations
- System Level Optimizations
- Experiments, Results
- Related Work
- **Conclusions and Future Work**

Conclusions & Future Work

- Integrated low-level hardware optimizations with high level middleware adaptations
- Studied multi-user environments
- Improved user experience and overall system functionality

- Future directions:
 - Further architectural optimizations (frame restructuring)
 - Middleware optimizations
 - System level adaptations, mobility factors

Thank you!