Sensing User Intention and Context for Energy Management

Angela Dalton
Carla Ellis

The Milly Watt Group
Duke University Department of Computer Science
Outline

- Motivation
- Case Study
  - FaceOff Architecture and Prototype
  - Evaluation
    - Best Case Feasibility Study
    - Responsiveness Study
- Future Work
- Conclusion
Motivation

- Current energy management techniques tied to process execution
- Can we use low power sensors to match I/O behavior more directly to user behavior and reduce system energy consumption?

Sensing User Intention and Context for Energy Management
Case Study: FaceOff

Displays:

- Typically responsible for large power drain
- Power State can be controlled by software
- State transition strategies naïve

A display is only necessary if someone is looking at it.
Prototype

**IBM ThinkPad running Linux**
- Base Power Consumption = 9.6 Watts
- Max CPU = 8.5 Watts over Base
- Display = 7.6 Watts

**Logitech QuickCam Web Cam**
- Power Consumption = 1.5 Watts

**Software components:**
- Image capture, skin detection, display power state control
Best Case Feasibility Study

What is the potential for energy savings?
- Assume Zero Overhead and Perfect Accuracy

Tradeoff of energy costs:
- CPU/Camera vs. Display

Effect on System Performance
- Network file transfer (113 MB)
- CPU intensive process (Linux kernel compile)
- MP3 Song (no display necessary)
File Transfer Traces

Power Trace for Large Network Transfer

- Without FaceOff
- With Faceoff

Power (Watts) vs. Time (Seconds)
Kernel Compile Traces

Power Trace for CPU Intensive Application

- Without FaceOff
- With FaceOff

Power (Watts)

Time (Seconds)
## Energy and Time Comparisons

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>With FaceOff</th>
<th>% Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Energy (J)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile</td>
<td>12506.85</td>
<td>11023.07</td>
<td>11.86</td>
</tr>
<tr>
<td>Transfer</td>
<td>6795.42</td>
<td>4791.19</td>
<td>29.49</td>
</tr>
<tr>
<td>MP3 Song</td>
<td>4,714</td>
<td>3,403</td>
<td>27.81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>With FaceOff</th>
<th>% Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time (s)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compile</td>
<td>575</td>
<td>603.5</td>
<td>4.96</td>
</tr>
<tr>
<td>Transfer</td>
<td>348.6</td>
<td>351.3</td>
<td>0.77</td>
</tr>
<tr>
<td>MP3 Song</td>
<td>251</td>
<td>251</td>
<td>No noticeable effect on playback</td>
</tr>
</tbody>
</table>

No noticeable effect on playback.
Responsiveness Study

- Use full prototype including skin detection
- Establish baseline timing
- Examine Responsiveness
  - varying system load
  - varying polling rate
Responsiveness Timing

Face arrives (or departs) → polling latency → Image acquired → detection latency → detection complete display signaled

Total responsiveness latency
Baseline Timing

Prototype Polling Latency
- On average _ image polling rate
- 500 ms on average for 1 s polling rate

Baseline Detection Latency
- ~135 ms
- Ran system for a period of one hour
- No load on system
## Detection Latency Under Load

<table>
<thead>
<tr>
<th>Workload</th>
<th>Average (99% Confidence)</th>
<th>Maximum</th>
<th>Minimum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network Transfer</td>
<td>175±7ms</td>
<td>305ms</td>
<td>116ms</td>
</tr>
<tr>
<td>Kernel Compile</td>
<td>230±5ms</td>
<td>669ms</td>
<td>51ms</td>
</tr>
<tr>
<td>MP3 Song</td>
<td>154±3ms</td>
<td>229ms</td>
<td>84ms</td>
</tr>
</tbody>
</table>
Varying Polling Rate

- Reduce overhead by reducing polling rate
  - Increases responsiveness latency

- Adaptive polling rate
  - Eliminate polling in presence of UI events
  - Begin polling as duration without UI events increases and face is detected
  - Reduce polling when no face present
    - Similar problem with latency increase upon return
Optimization with Motion Sensor

- Combine adaptive polling & motion sensing
- Meet responsiveness requirements with minimal FaceOff system overhead
- Eliminate image polling when no motion
- Switch display state on immediately when motion detected and restart image polling
Implementation

Prototype using X10 ActiveHome Wireless Motion Sensor and Receiver

- Receiver connects to serial port
- Reading port blocks until sensor triggers
- Takes up to 10 seconds to recharge

Promising addition to FaceOff system
More Roles for Sensors

- **Touch Sensor**
  - Detect picking up of a PDA

- **Light, Sound sensors**
  - Adjust display brightness (iPAQ)
  - Adjust speaker volume

- **802.11 Signal Strength sensor**
  - Determine possibility of offloading computation
Enhanced Sensors

“Active Camera”
- Perform some or all of the face detection

Color filtering
- Preprocessing skin color segmentation

Low Power microcontroller for external sensor control, computation
Future Work

- Continue work on optimizing responsiveness

- Comprehensive user study
  - Survey of usability
  - Characterization of usage patterns
    - End-to-end experiment
Conclusions

Context information offers promising method of energy management

FaceOff illustrates feasibility of approach

Available very low power sensors as well as optimization techniques would improve upon the FaceOff energy savings
Questions?

The Milly Watt Group
http://www.cs.duke.edu/ari/millywatt
Duke University Department of Computer Science