The Illinois GRACE Project:

Cross-Layer Adaptation for Saving Energy

Faculty: **Sarita V. Adve, Douglas L. Jones, Robin H. Kravets, Klara Nahrstedt**

Students: **Albert F. Harris, Won Jeon, Daniel Grobe Sachs, Vibhore Vardhan**

Alumni: **Christopher J. Hughes, Wanghong Yuan**

Computer Science and Electrical & Computer Engineering

University of Illinois at Urbana-Champaign

grace@cs.uiuc.edu

Motivation

Target system

- Mobile devices w/ multimedia apps, wireless communication

Challenges

- Stringent, dynamic, multidimensional resource constraints

Computation	**Energy**	**Network bandwidth**
Real-time	**Battery life**	**Losses, congestion**

Opportunities

- **Real-time** ⇒ Needn't go faster than real-time
- **Dynamic** ⇒ Slow down to save energy   many current processors!
- **Soft correctness** ⇒ Trade output quality for resource use

Key Observations

Dynamic resources & demands + Flexible output quality ⇒

Use adaptation to respond to changes

Adapt all system layers

Hardware, network, operating system, application, …

All layers must adapt cooperatively

to maximize user experience - system utility

while meeting current resource constraints

⇒ GRACE – Global Resource Adaptation through CoopEration

Example for Cross-Layer Adaptation

Consider real-time video delivery over wireless network

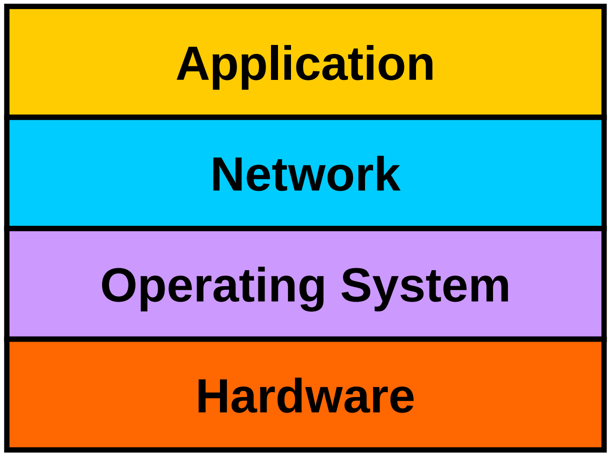
Application **How much compression?** **What quality?**
Network **How much transmit power?** **How much error correction for wireless channel?**
Operating System **How to allocate resources to multiple applications?** **How to allocate among components of the same application?**
Hardware **Which frequency, voltage?** **Which processor, cache, memory configuration?**

Each adaptive layer must make several decisions affecting

- all resources – CPU time, network bandwidth, system energy
- output quality
- other layers

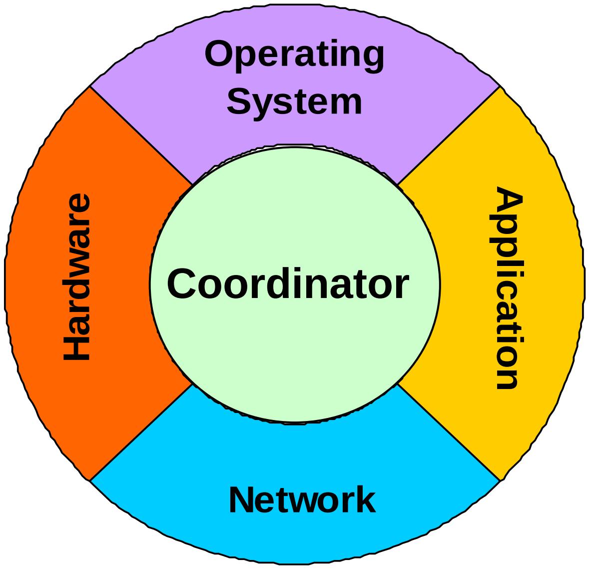
Current Practice vs. GRACE

Current practice



- System divided into layers
- Adapt 0, 1, or 2 layers in isolation

GRACE



- Global community
- All adapt cooperatively
- Retain advantages of layering with clean, minimal *interfaces*

Overview

Challenges in Cross-Layer Adaptation

GRACE Adaptation Hierarchy

GRACE System Layers and Adaptations

Putting it Together

Experimental Testbed and Results

Conclusions and Future Work

Challenges in Cross-Layer Adaptation (1 of 2)

	What to adapt?	**When to adapt?**	
Ideal:	All layers, all apps	Frequent	Expensive
Practical?:	All layers, all apps	Infrequent	Imprecise
	One app, one system layer	Frequent	Limited scope

GRACE solution = ***hierarchical adaptation***

Three adaptation levels: global, per-app, internal

- global: **full system,** infrequent
- per-app, internal: **frequent,** limited scope

Challenges in Cross-Layer Adaptation (2 of 2)

Implementing cross-layered hierarchical adaptation is difficult

- Multiple adaptations
- Multiple time scales
- What information to expose at each layer?
- How and when to communicate information between layers?
- $\Rightarrow$ Interfaces need to be well designed

Overview

Challenges in Cross-Layer Adaptation

GRACE Adaptation Hierarchy

- Global
- Per-app
- Internal

GRACE System Layers and Adaptations

Putting it Together

Experimental Testbed and Results

Conclusions and Future Work

GRACE Components

Adaptive layers

CPU, network, application, scheduler

- Each layer has multiple configurations

500MHz, 750MHz, 1GHz

- Each configuration impacts resource usage, app quality

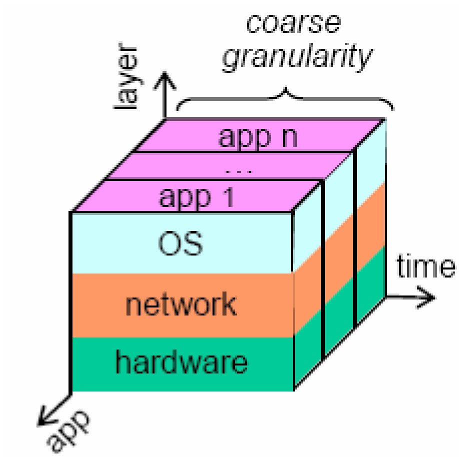
1s, 1J   0.8s, 2J   0.5s, 4J

Adaptation controllers

- Choose configuration for a layer
  - ⇒ Choose resource usage, app quality
- Global, per-app, internal controllers

Global Adaptation (1 of 2)

Adapts all applications and system layers at large changes



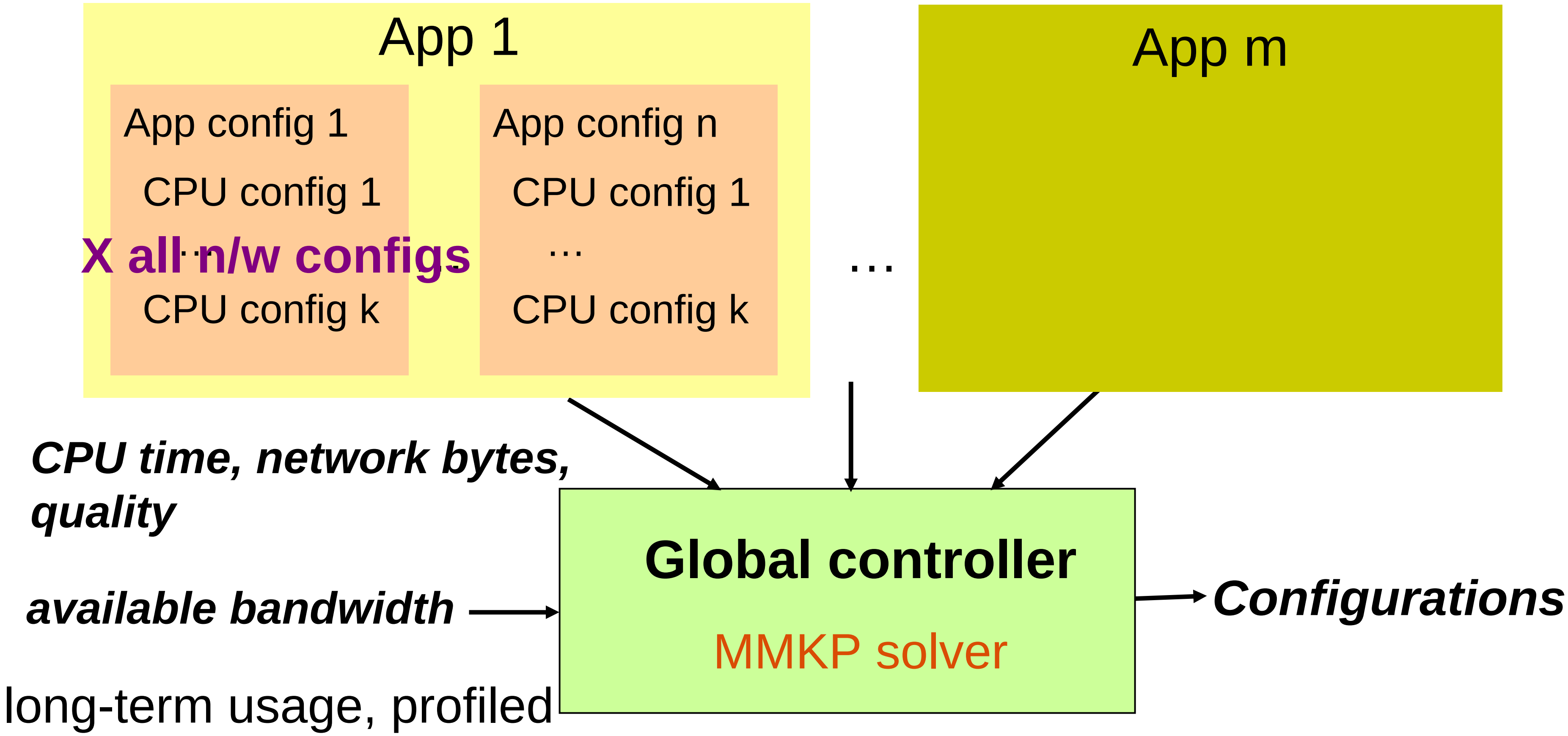
Goal: For all apps,

choose app, CPU, network, … configuration such that

- optimize objective function minimize CPU + n/w energy
- subject to constraints CPU time, n/w bandwidth, app quality

MMKP problem - expensive

Global Adaptation (2 of 2)

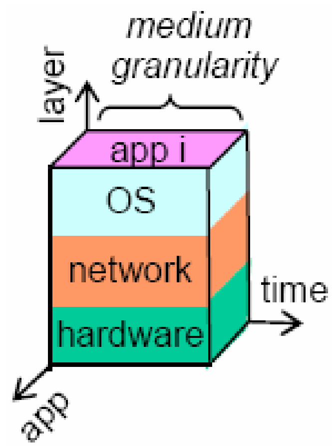


Expensive – triggered on large changes (e.g., app entry, exit)

Adapts for long-term resource demands and availability

Per-Application Adaptation (1 of 2)

Considers one application at a time - adapts all layers



Global adaptation decision
= resource allocation

Triggered at granularity meaningful to application (e.g., frame)

Adapts for resource demand for next frame

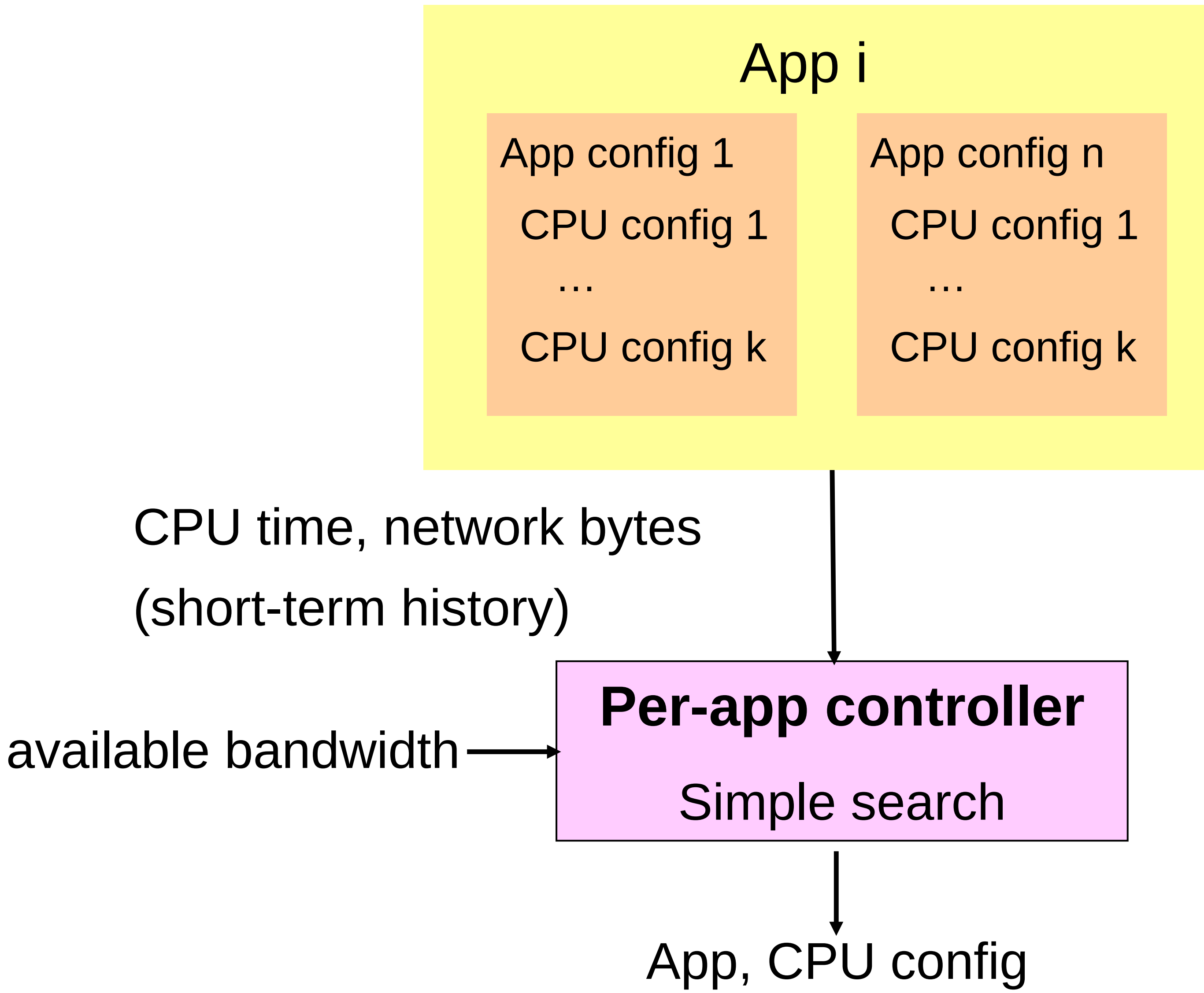
Goal: For a single app,

choose app, CPU, network configuration such that

- minimize CPU + network energy (for next frame)
- subject to CPU time, bandwidth, app quality constraints

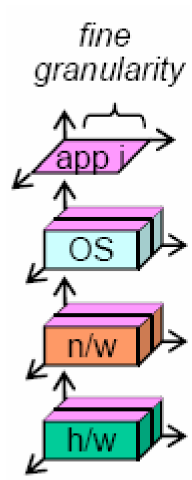
allocation from global

Per-app Adaptation (2 of 2)



Internal Adaptation

Adapts single system layer



Triggered at granularity meaningful to layer

E.g., packets for network

Respects resource allocation from global

Not visible to rest of the system

Overview

Challenges in Cross-Layer Adaptation

GRACE Adaptation Hierarchy

GRACE System Layers and Adaptations

- CPU
- Network
- Application
- O.S. Scheduler

Putting it Together

Experimental Testbed and Results

Conclusions and Future Work

Current GRACE Prototype

Adaptive CPU, network, application, OS scheduler

Pentium M
laptop

IEEE 802.11b

Earliest deadline first (EDF)
soft real-time scheduler

H.263 based video encoder

Other non-adaptive apps

Periodic (e.g., video frames)

The CPU Layer

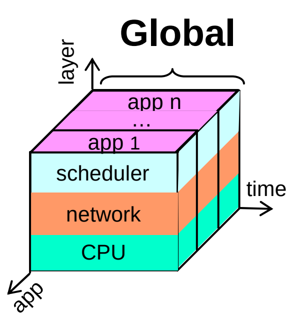
Adaptation mechanisms

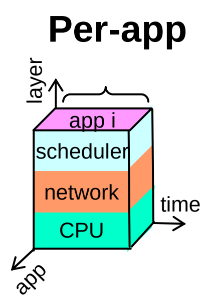
Dynamic voltage and frequency scaling (DVFS)

Impact

Energy ↓ Execution time ↑

Control hierarchy





Layer-specific details

Decisions based on dynamic power $\Rightarrow$ Slower is better

The Network Layer

Adaptation mechanisms

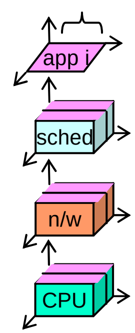
Transmit power, bandpass modulation retransmission

Impact

Energy ↓ Data rate ↑ Reliability?

Control hierarchy

Internal



Layer-specific details

IEEE 802.11b $\Rightarrow$ Faster is better

The Application Layer

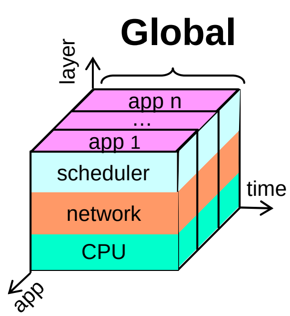
Adaptation mechanisms

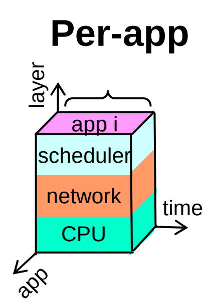
- Trade off CPU work for amount of compression
  - Terminate motion search early
  - Eliminate some discrete-cosine transforms (DCT)
  - I-frames – force use, send portions uncoded

Impact

Energy **?** CPU work ↓ Bandwidth ↑ Quality similar

Control hierarchy





The OS Scheduler Layer

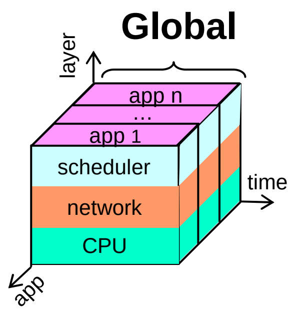
Adaptation mechanisms

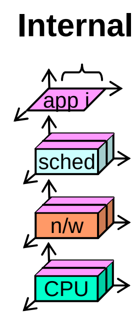
- Allocation of CPU time among applications

Impact

- Energy, CPU time, deadline misses

Control





Layer-specific details

- Monitors CPU budget, reclaims unused CPU budget

Overview

Challenges in Cross-Layer Adaptation

GRACE Adaptation Hierarchy

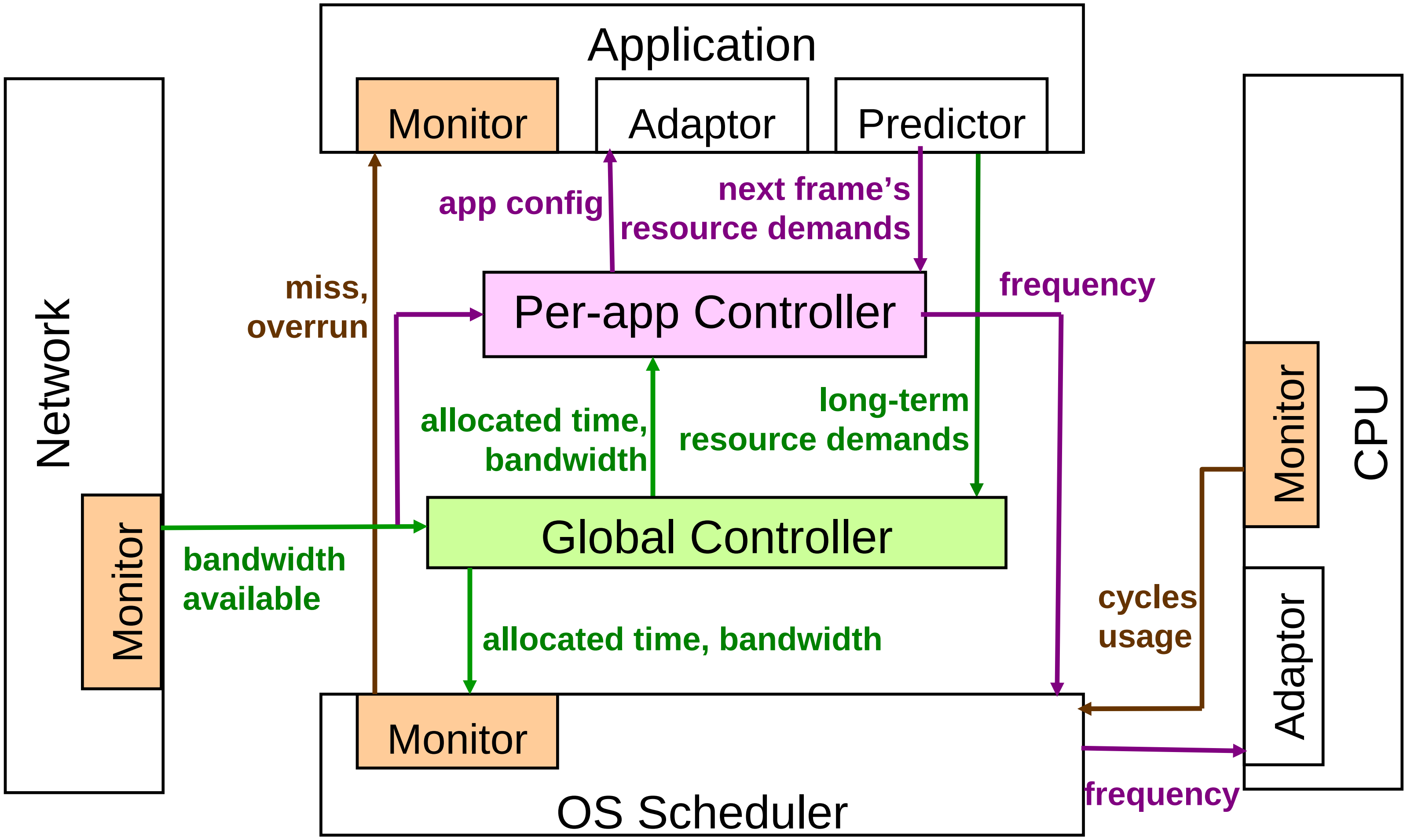
GRACE System Layers and Adaptations

Putting it Together

Experimental Testbed and Results

Conclusions and Future Work

GRACE System Architecture



Overview

Challenges in Cross-Layer Adaptation

GRACE Adaptation Hierarchy

GRACE System Layers and Adaptations

Putting it Together

Experimental Testbed and Results

Conclusions and Future Work

Experimental Testbed

Two laptops: One GRACE, one non-GRACE

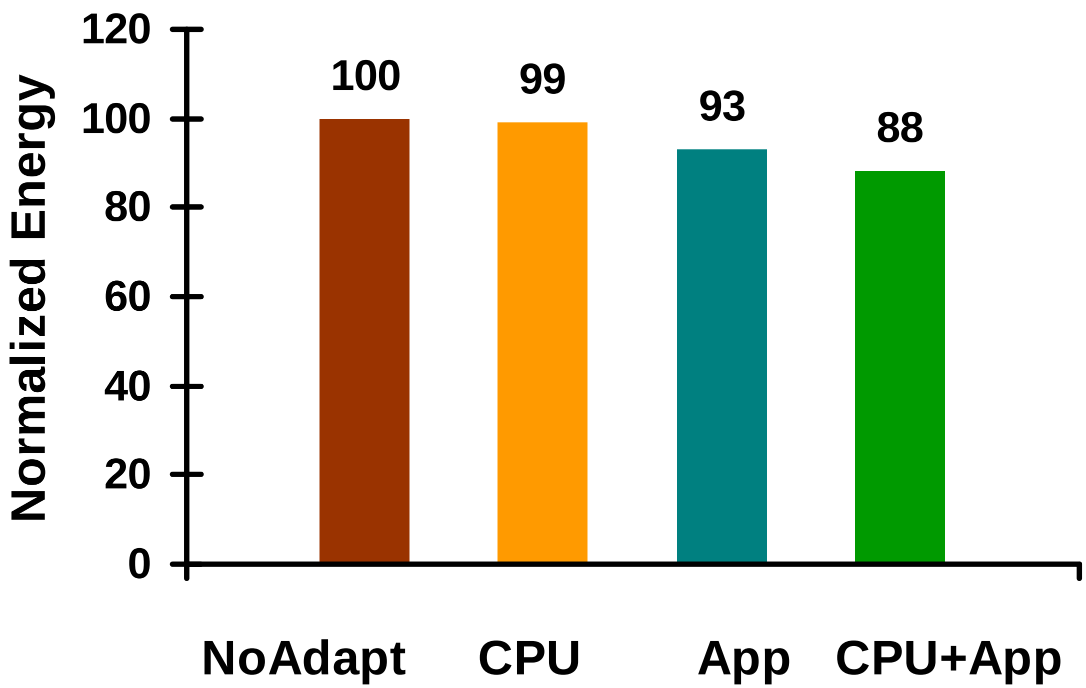
Run video conference (no audio) over IEEE 802.11b ad hoc link

- Video encoder (adaptive), decoder (non-adaptive)
- 10fps, 320x240 video from web cam
- Cisco Aironet wireless card

Power meter measures system watts for GRACE laptop

Results From Prototype

Canned video stream for (partial) repeatability



GRACE gives 12% energy savings for entire system

App + CPU adaptation better than sum of each alone

Simulation Results - Methodology

Simulation/emulation for repeatable experiments

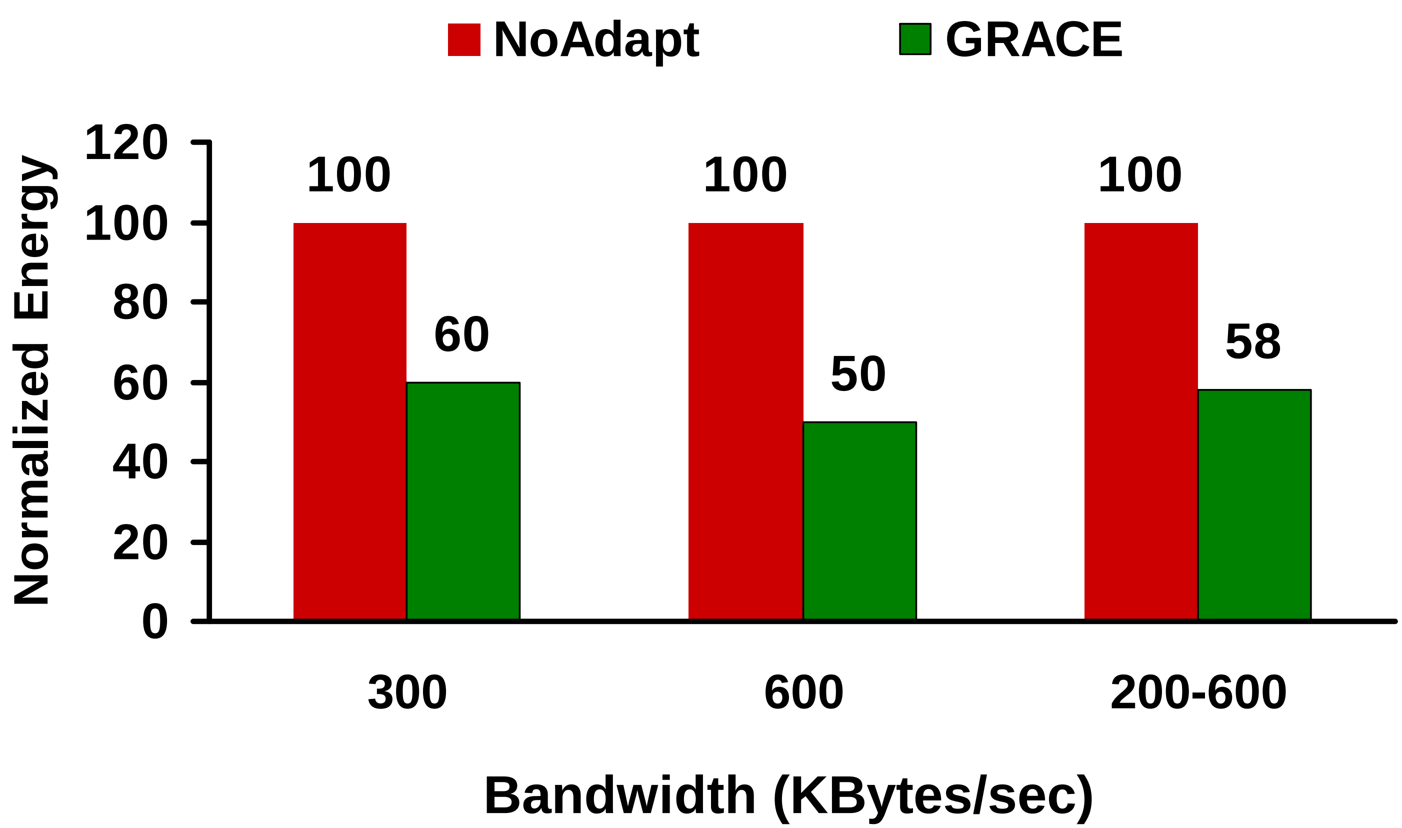
- CPU: Athlon Mobile XP 1700+, up to 25W
- Network: 750mW active power, 3 bandwidth models
  - Fixed constrained – 200 Kbytes/s
  - Fixed unconstrained – 600 Kbytes/s
  - Variable – 200 to 600 Kbytes/s

Workload

- Four encoders
- Canned video streams
- QCIF, 15fps

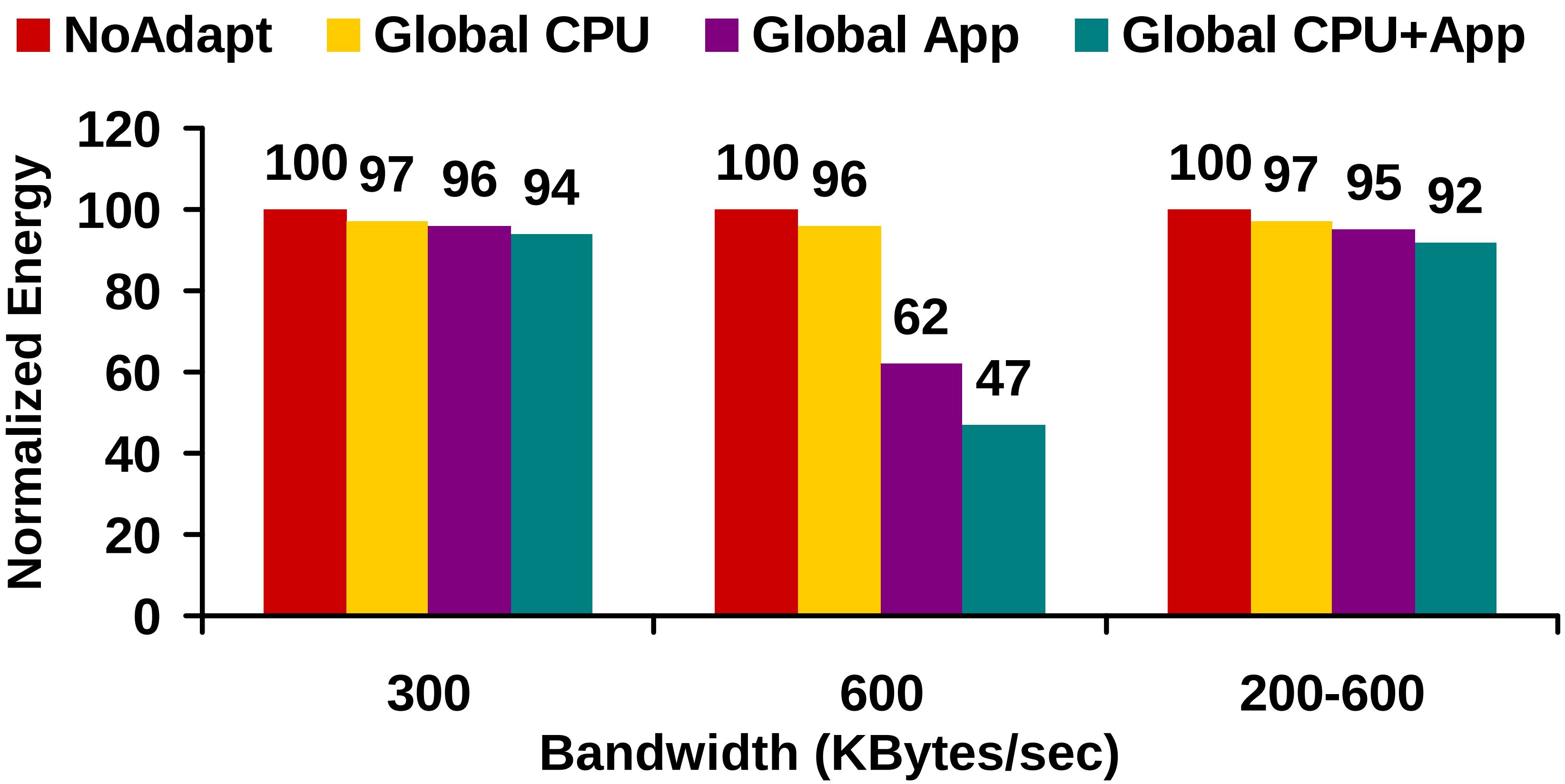
Energy reported only for CPU+network

Overall Results



GRACE provides 40% to 50% energy savings over large range

Benefits of Global Adaptation

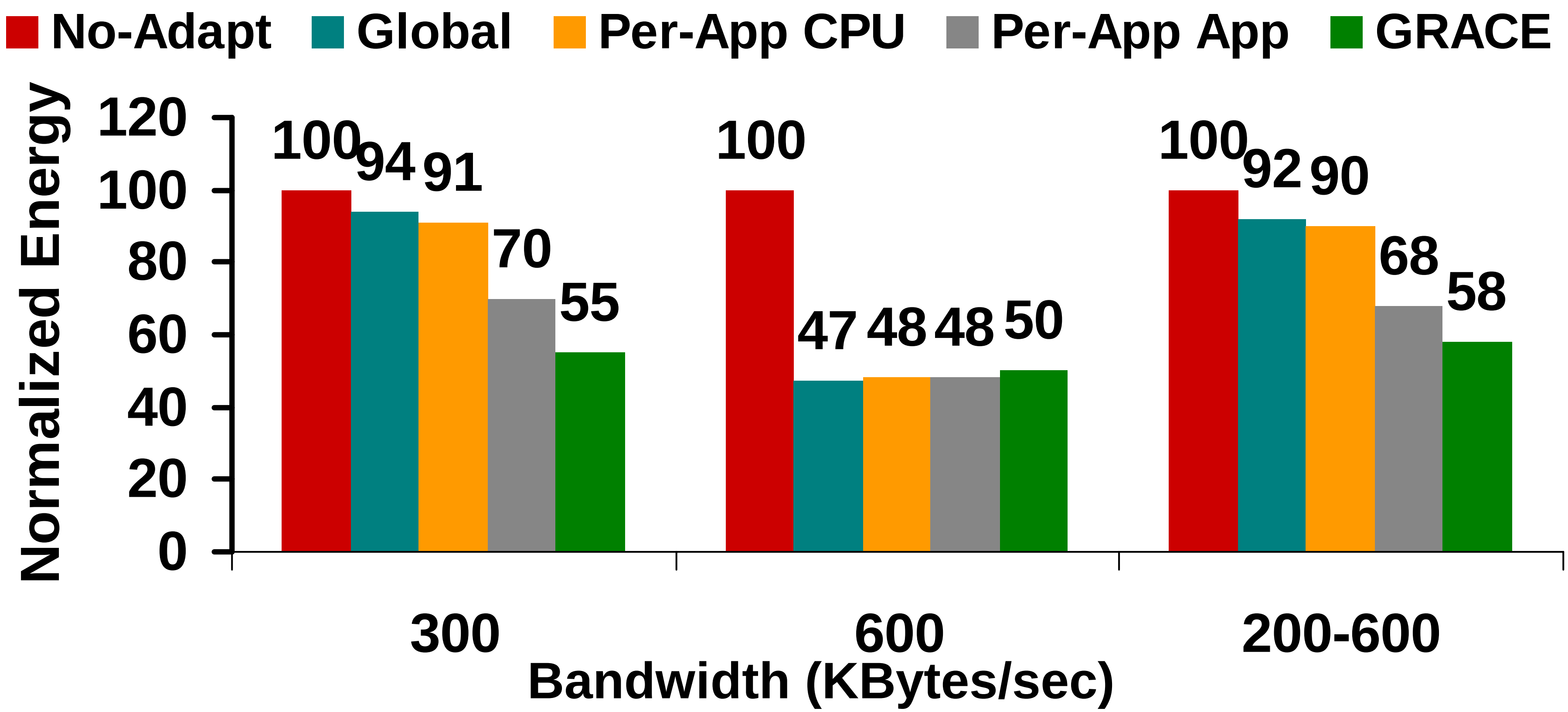


Global works well with fixed, unconstrained bandwidth

App adaptation gives most benefits

App + CPU better than sum of each alone

Benefits of Per-App Adaptation

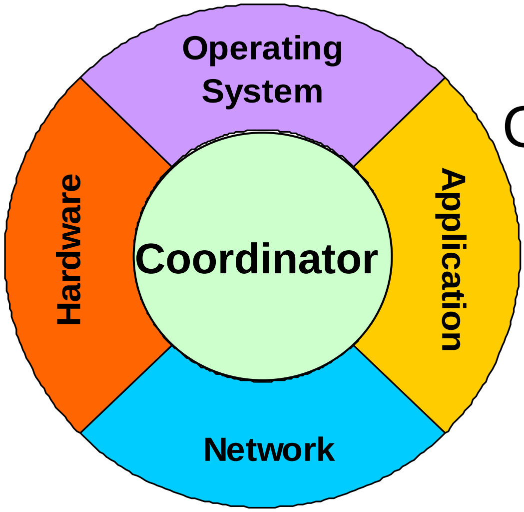


Per-app adaptation responds to constraints, bandwidth variations

Application adaptation gives most benefits

App + CPU better than sum of each alone

Conclusions



GRACE – Saving energy for mobile multimedia

- Cross-layer, hierarchical adaptation
- Frequent, multi-layer, practical adaptation

Real implementation

- Adaptive CPU, network, app, scheduler
- Global, per-app, internal adaptation
- Significant benefits from all layers, all adaptation levels

Saved 12% system energy (real), 50% CPU+n/w energy (model)

Future Work

Reliability

Other objective functions for global adaptation - utility

Other networks; e.g., cellular

Application adaptations and predictions

OS scheduling for application groups

Integrating architecture adaptations (multicore, heterogeneity)

Other layers – memory, disk, …

Distributed systems and new applications