The Illinois GRACE Project:

Global Resource Adaptation through CoopEration

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

Students: **Albert F. Harris, Christopher J. Hughes, Daniel Grobe Sachs,**

Ruchira Sasanka, Jayanth Srinivasan, 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

New challenges

- Stringent, dynamic, multidimensional resource constraints

Computation	**Energy**	**Bandwidth**
Real-time	**Battery life**	**Losses, congestion**

New opportunities

- **Real-time and dynamic** ⇒ Slow processing to save energy
- **Soft correctness** ⇒ Trade output quality for resource use

Key Observations

Dynamic resource constraints + Flexible output quality ⇒

Use adaptation to respond to changes

Adapt all system layers

Hardware, network, operating system, application, …

All layers must adapt cooperatively

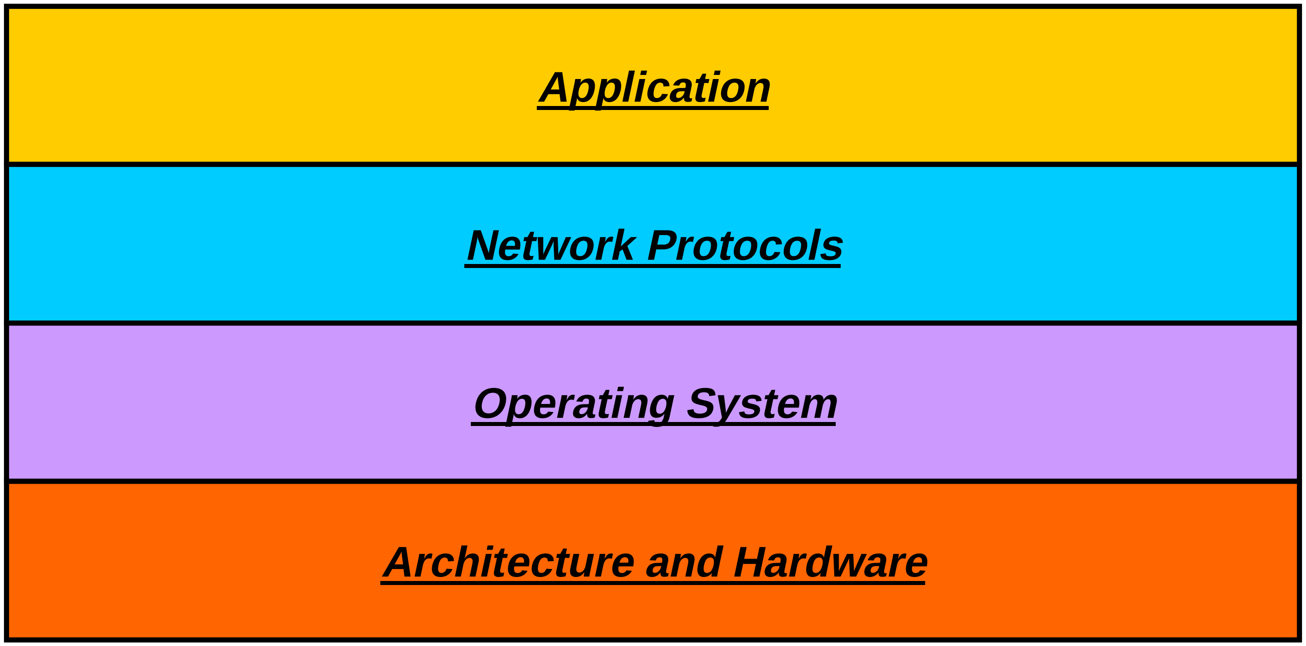
to maximize user experience - system utility

while meeting current resource constraints

⇒ GRACE – Global Resource Adaptation through CoopEration

Example for Cross-Layer Adaptation

Consider real-time video delivery over wireless & wired network

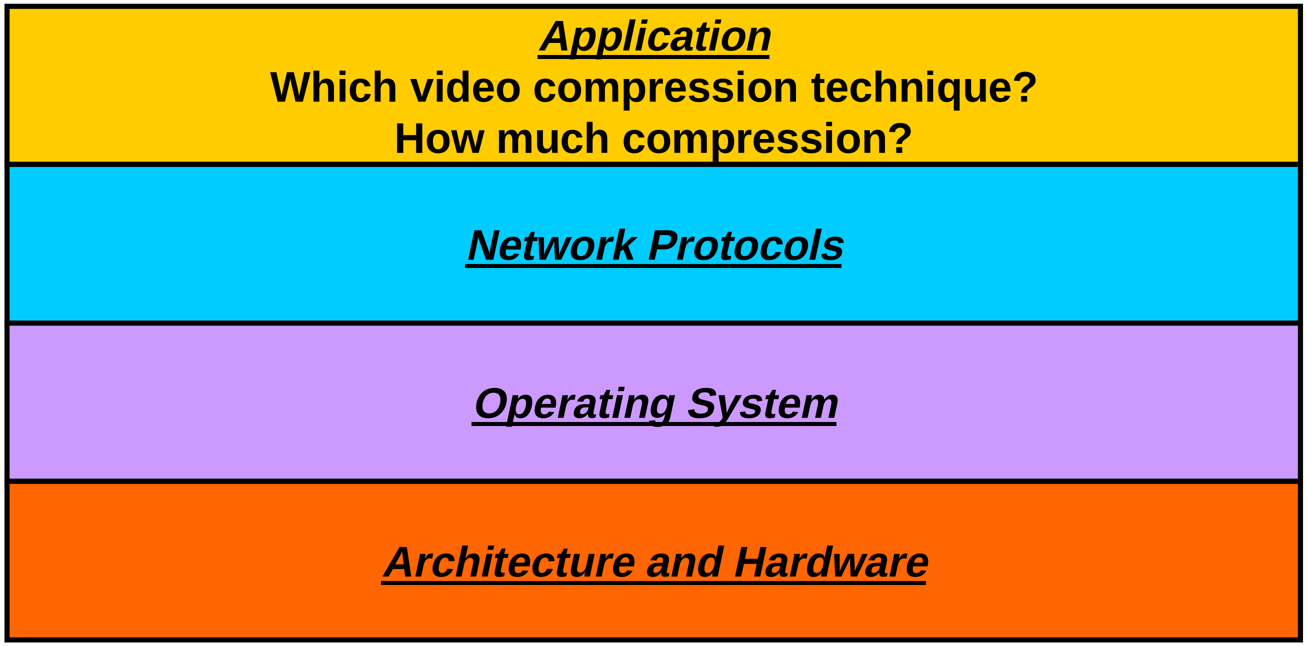


Each adaptive layer must make several decisions affecting

- all resources - time, energy, bandwidth
- other layers

Example for Cross-Layer Adaptation

Consider real-time video delivery over wireless & wired network



Each adaptive layer must make several decisions affecting

- all resources - time, energy, bandwidth
- other layers

Example for Cross-Layer Adaptation

Consider real-time video delivery over wireless & wired network

Application
Which video compression technique?
How much compression?

Network Protocols
How much error correction for wireless channel?
Which congestion control protocols for wired network?

Operating System

Architecture and Hardware

Each adaptive layer must make several decisions affecting

- all resources - time, energy, bandwidth
- other layers

Example for Cross-Layer Adaptation

Consider real-time video delivery over wireless & wired network

Application
Which video compression technique?
How much compression?

Network Protocols
How much error correction for wireless channel?
Which congestion control protocols for wired network?

Operating System
How to allocate resources to multiple applications?
How to allocate among components of the same application?

Architecture and Hardware

Each adaptive layer must make several decisions affecting

- all resources - time, energy, bandwidth
- other layers

Example for Cross-Layer Adaptation

Consider real-time video delivery over wireless & wired network

Application **Which video compression technique?** **How much compression?**
Network Protocols **How much error correction for wireless channel?** **Which congestion control protocols for wired network?**
Operating System **How to allocate resources to multiple applications?** **How to allocate among components of the same application?**
Architecture and Hardware **Which processor, cache, memory configuration?** **Which frequency, voltage?**

Each adaptive layer must make several decisions affecting

- all resources - time, energy, bandwidth
- other layers

State-of-the-Art

Most current work adapts single layer at a time

Some jointly adapt 2 layers, BUT one layer drives adaptation

E.g., app controls video coding and n/w error correction

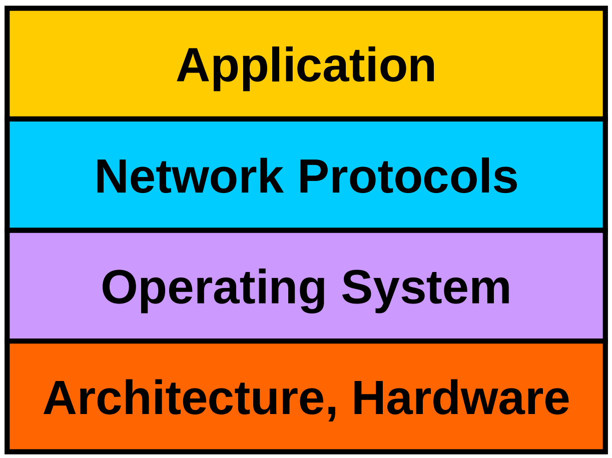
- Exposes internals of one layer to another
- Sub-optimal use of system flexibility
- Difficult to scale to more than two adaptive layers

Need new solutions that will

+ Retain software engineering advantages of layers
+ Exploit full system flexibility for globally optimal solution
+ Scale to multiple adaptive layers

Current Systems vs. GRACE

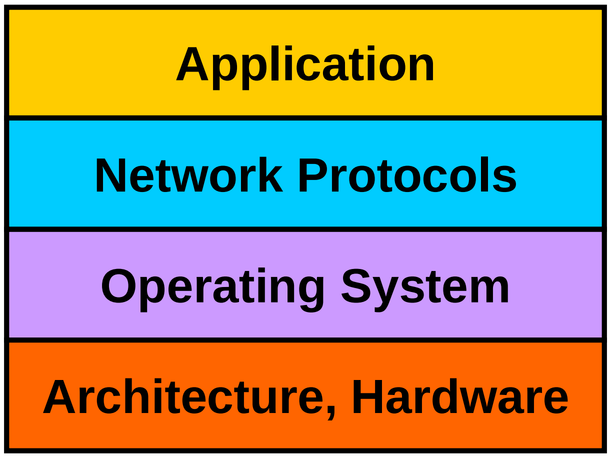
Current approaches



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

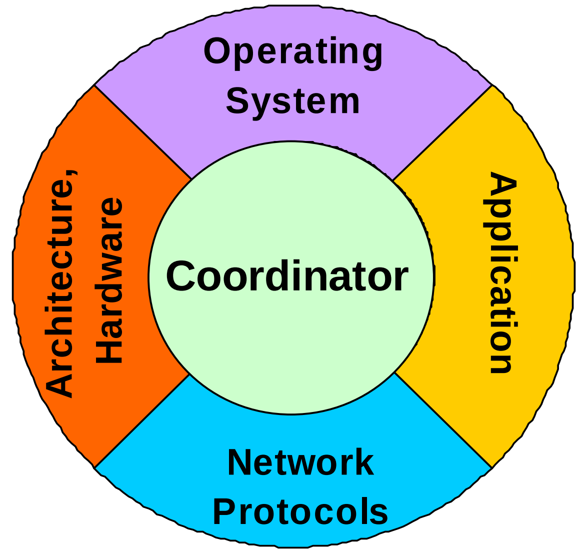
Current Systems vs. GRACE

Current approaches



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

GRACE



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

GRACE Framework - Overview

Two major adaptation modes

Global | **Local**

GRACE Framework - Overview

Two major adaptation modes

Global

- Via resource manager - RM
- Expensive
- Triggers: rare, coarse-grain
  - Application arrives, leaves
  - Large resource change
  - Large usage change
- RM reallocates resources to apps to maximize system utility

Local

GRACE Framework - Overview

Two major adaptation modes

Global

- Via resource manager - RM
- Expensive
- Triggers: rare, coarse-grain
  - Application arrives, leaves
  - Large resource change
  - Large usage change
- RM reallocates resources to apps to maximize system utility

Local

- Individual layers adapt locally
- Cheap
- Triggers: frequent, fine-grain
  - Small change in resource use
- Respect global allocation of resources, utility

Key Interfaces – Cost and Utility

All layers adapt locally

- No knowledge of internals of other layers
- Exposed information: cost and utility (of app configuration)

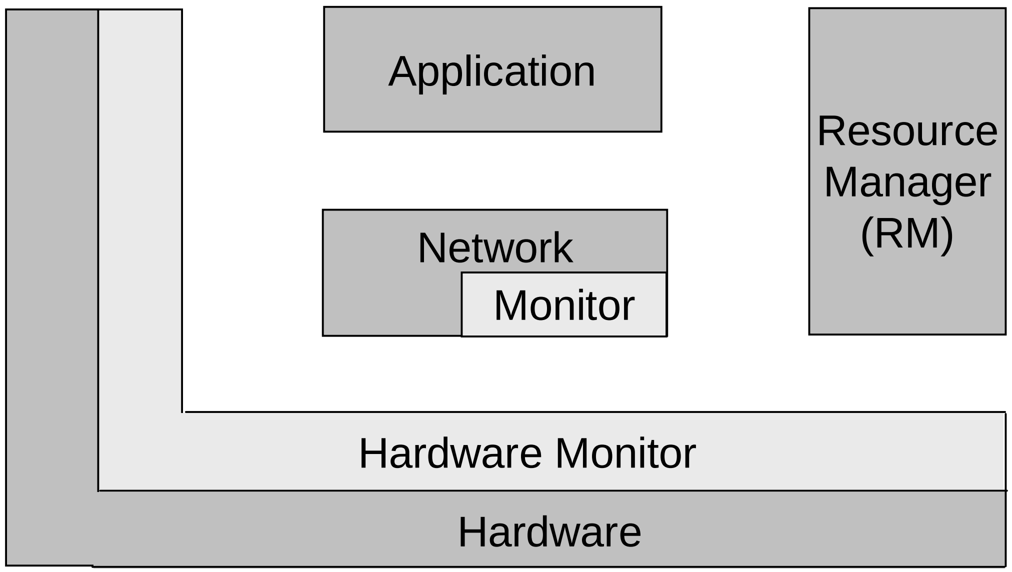
Cost (of an application configuration)

- Computation time, energy, bandwidth/reliability
- From hardware, other software components (e.g., network)
- Multiple operating points (costs) for each resource
- Get from dynamic profiling, programmer, compiler

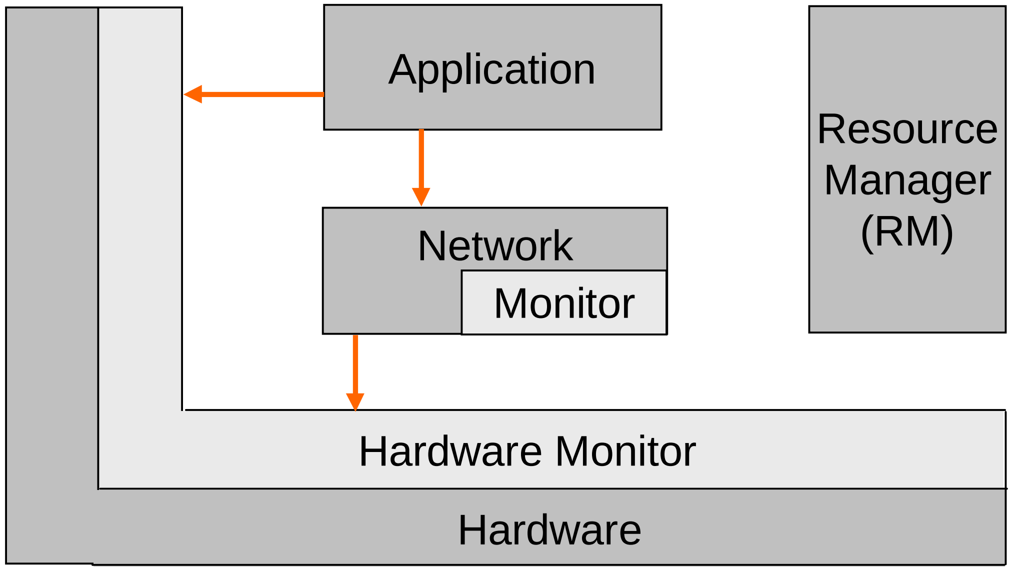
Utility (of an application configuration)

- Depends on QoS level, importance of application

Communication in GRACE



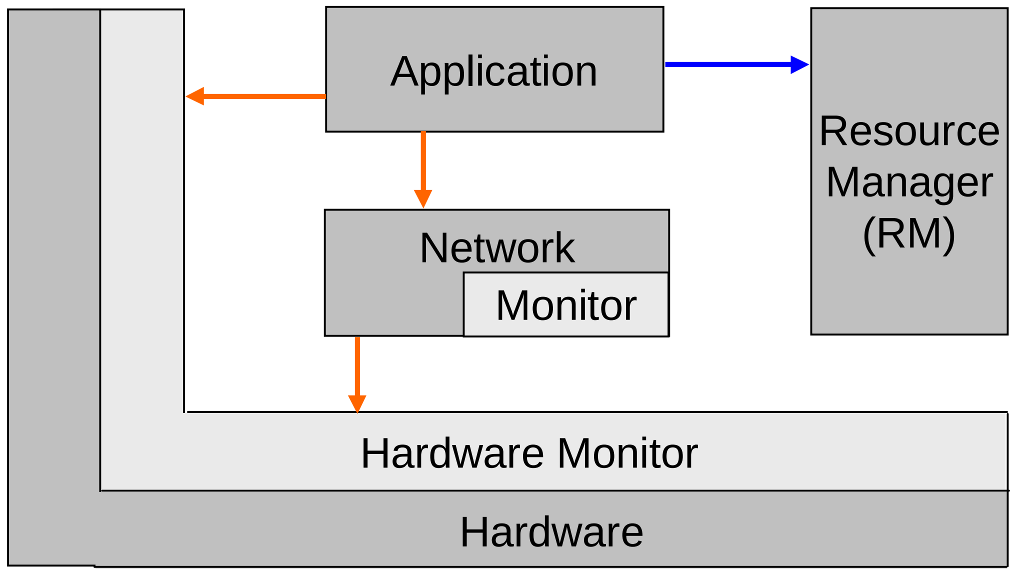
Communication in GRACE



Show only requests

Cost: App queries hardware, network for cost of each configuration

Communication in GRACE

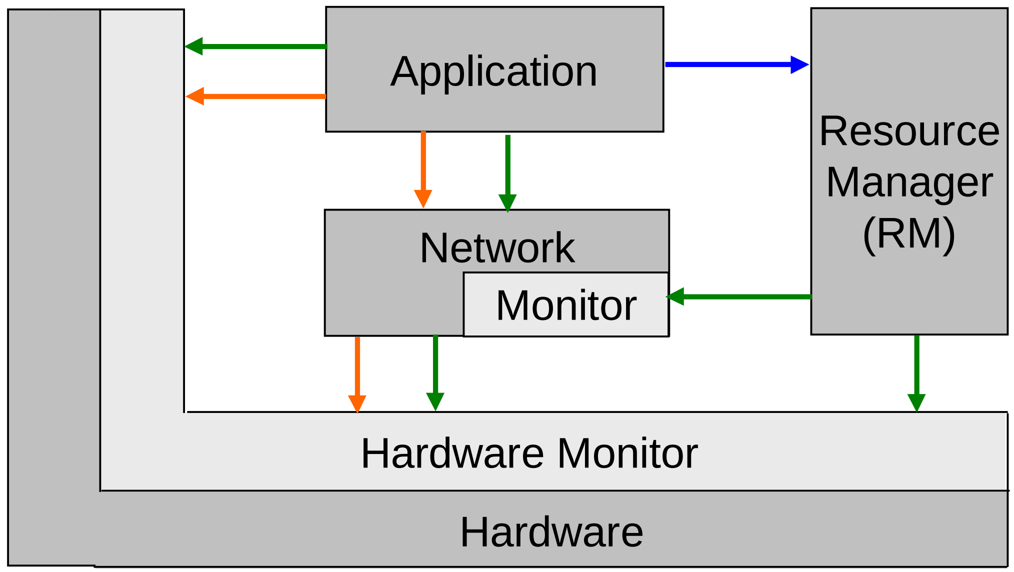


Show only requests

Cost: App queries hardware, network for cost of each configuration

Reserve: App requests reservation with (cost, utility) options

Communication in GRACE



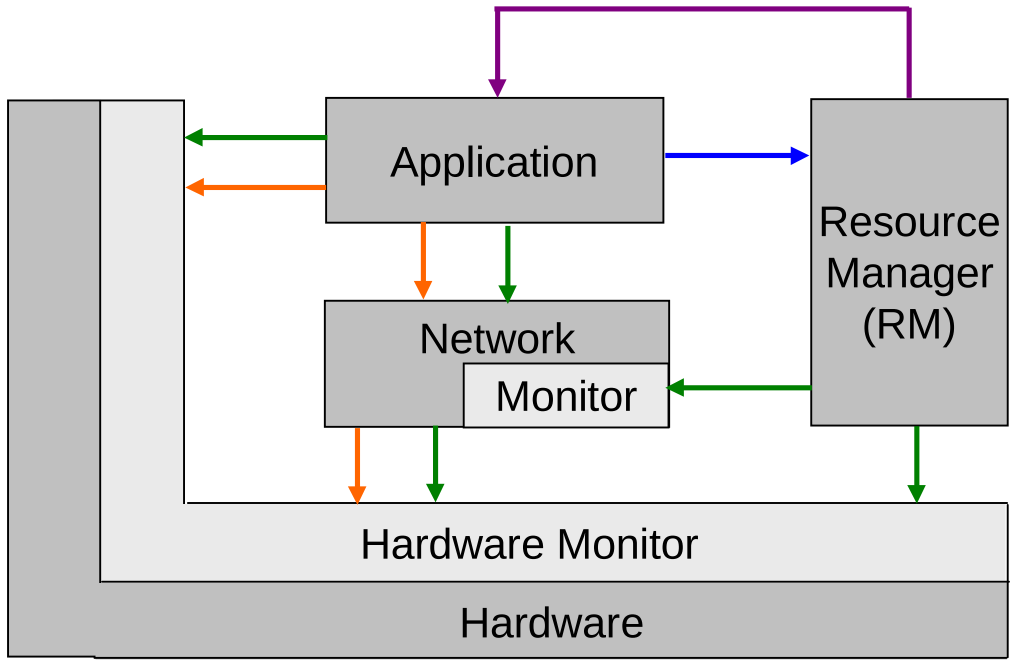
Show only requests

Cost: App queries hardware, network for cost of each configuration

Reserve: App requests reservation with (cost, utility) options

Monitor: App, RM monitor resource use (cost) for local adaptation

Communication in GRACE



Show only requests

Cost: App queries hardware, network for cost of each configuration

Reserve: App requests reservation with (cost, utility) options

Monitor: App, RM monitor resource use (cost) for local adaptation

Update: RM triggers reconfiguration at global change

Global Adaptation

Example: New application enters system

Global Adaptation

Example: New application enters system

Determine cost of new app for highest utility

Global Adaptation

Example: New application enters system

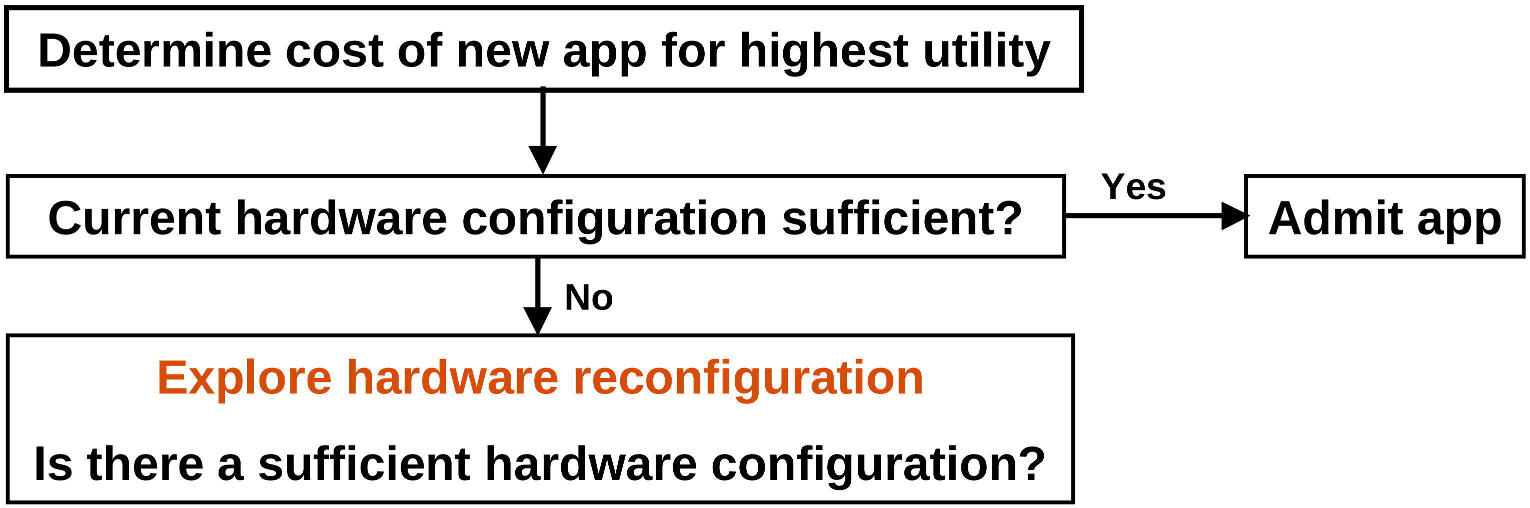
Determine cost of new app for highest utility

↓

Current hardware configuration sufficient?

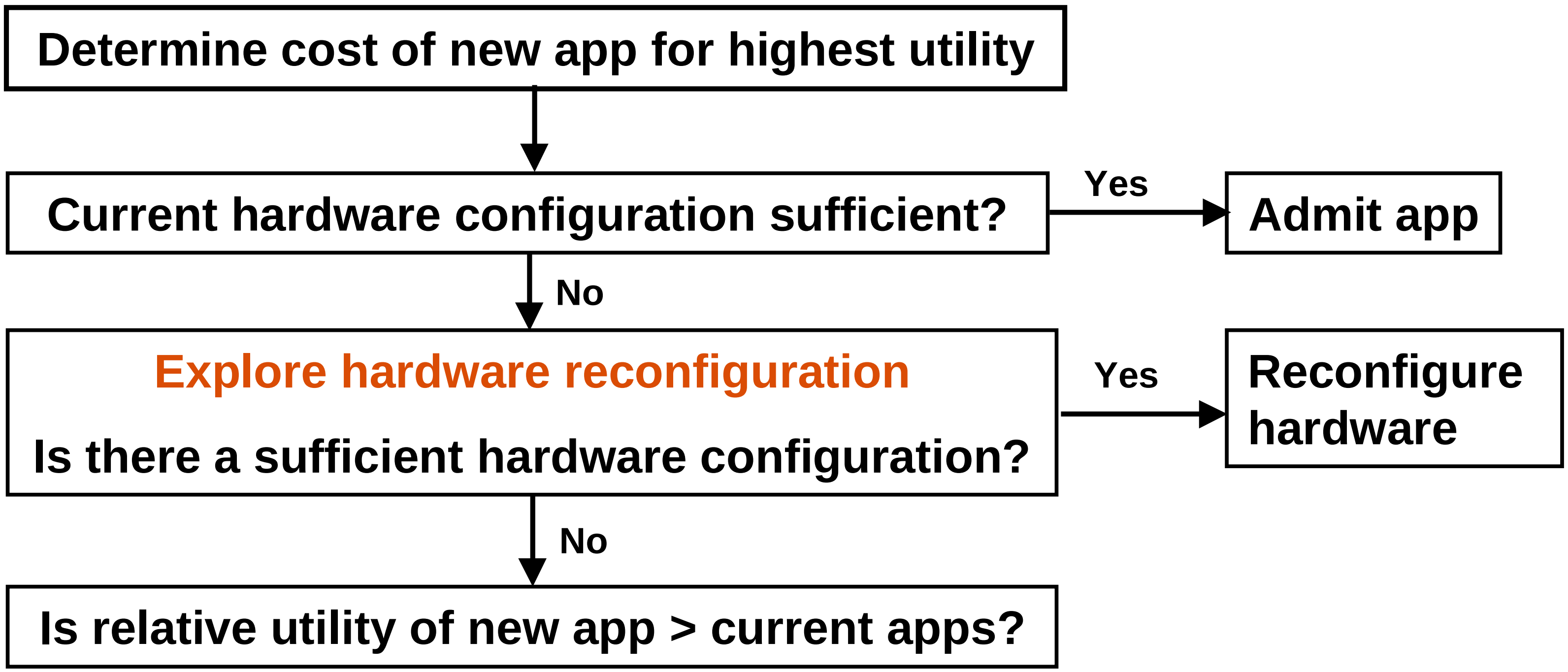
Global Adaptation

Example: New application enters system



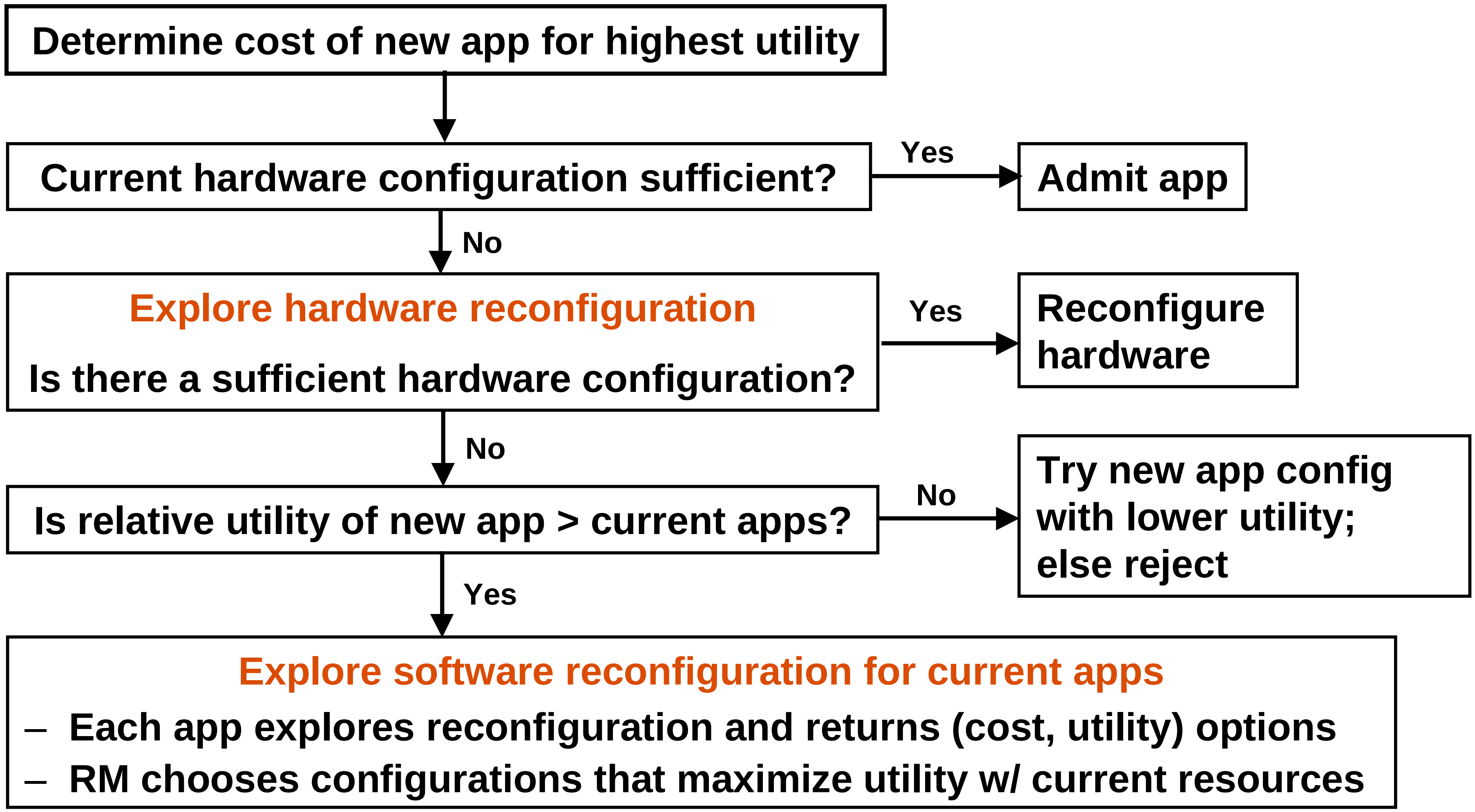
Global Adaptation

Example: New application enters system



Global Adaptation

Example: New application enters system



Local Adaptation

Each layer free to adapt locally after global resource allocation

- Must respect global allocation, utility
- Adaptation process specific to the layer, oblivious to others

If local adaptations consistently produce lower resources

Resource manager triggers new global adaptation

Experience and Initial Results

Local hardware adaptation

Local application adaptation

Resource Manager + Hardware + Application adaptations

Local Hardware Adaptation

Multiple levels of adaptation [Hughes et al. Micro'01, Sasanka et al. Asplos'02]

- Dynamic voltage (and frequency) scaling (DVS)
- Architecture adaptation

  E.g., # active functional units, instruction window size

  – Coarse-grain inter-frame, fine-grain intra-frame

  – Inter picks max config for frame, intra adapts within max

Significant energy savings for apps/systems studied

82% vs. no adaptation, 28% vs. only DVS, 66% vs. only arch

Local Application Adaptation

Initial adaptation for video encoder

- Dynamic adjustment of escape threshold for motion search
- Affects CPU & transmission energy, but not output quality
- 6% energy benefit in one case, on adaptive hardware

Experience with Resource Manager

- With only adaptive hardware through DVS – simulation

  [Yuan & Nahrstedt, NOSSDAV'02]

- With DVS and adaptive applications – real implementation

  [Yuan et al., submitted for publication]

Resource Manager + DVS + Adaptive App

System with DVS and adaptive applications

- Global adaptation
  - Cost based on average computation time of a frame
  - Maximize utility given battery energy and needed lifetime
  - Triggers hardware and/or application adaptation
  - No network consideration yet
- Local adaptation
  - Only frequency adjustment to handle overrun, underrun
  - Application adaptation too expensive in system studied

Experimental Methodology

Implemented in Linux kernel on HP laptop with Athlon processor

Applications

- Multiple MPEG decoders - 3 options for frame rate, dithering
- Utility = monotonic function of processor utilization
  (more work $\Rightarrow$ higher utility)

Hardware: 6 frequency/voltage configurations on Athlon CPU

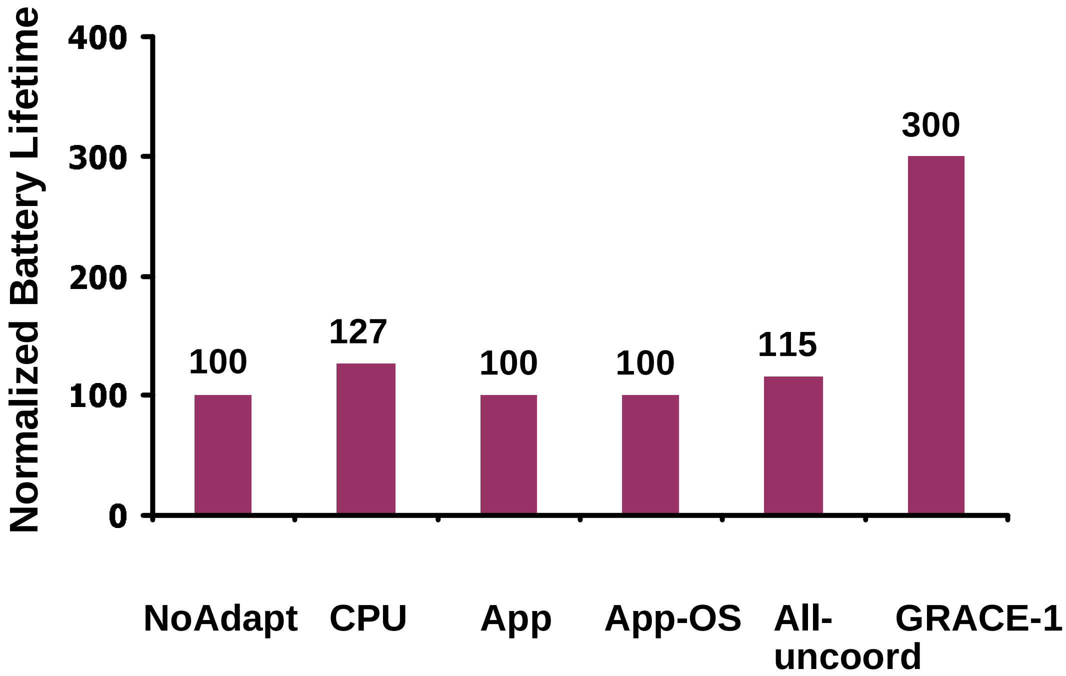
Metrics: Achieved battery lifetime, utility

Compared with No adapt, CPU-only, App-only, App-OS,
uncoordinated App-OS-CPU

Results assume available energy = 1/3 needed at peak config

Initial Results

Achieved battery lifetime (normalized) with different adaptations



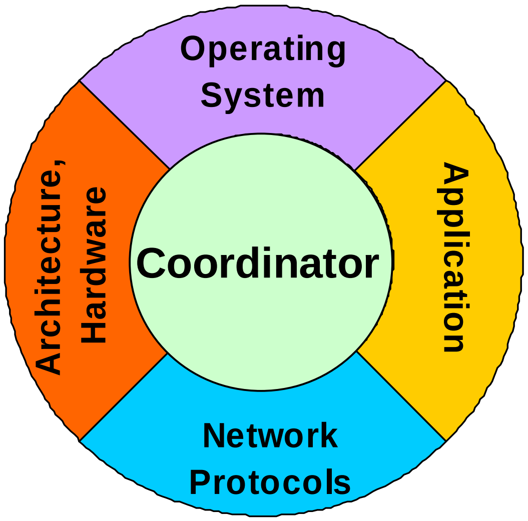
In all cases, utility for GRACE-1 was highest

Benefits reduce with more energy availability

Summary

Mobile systems with multimedia apps

GRACE – Global Resource Adaptation with CoopEration



All system layers adapt cooperatively

to maximize system utility

within available resources

- Mediated by resource manager – says *what*, not *how*
- Clean, minimal interfaces
- Retains advantages of layers, exploits full flexibility, scalable

Much remaining work to realize GRACE goals