Linux Dionisys: A Kernel-Based Approach to QoS Management

Richard West & Jason Gloudon

Operating Systems & Services Group

Computer Science

Motivation

Computer Science

- General purpose systems have limitations:
  - Ill-equipped to meet service requirements of complex real-time applications
- Aim to **extend** COTS systems to:
  - better meet the service needs of applications
  - provide finer-grained service management than at user-level
  - adapt system behavior to compensate for changes in resource needs and availability

Approach

- Linux Dionisys
  - Distributed system for run-time service adaptation
  - Allow real-time applications to specify:
    - how, when & where actual service should be adapted to meet required / improved QoS
  - MEDEA: Mechanism for Event DrivEn Adaptation
  - SafeX: Safe kernel eXtensions

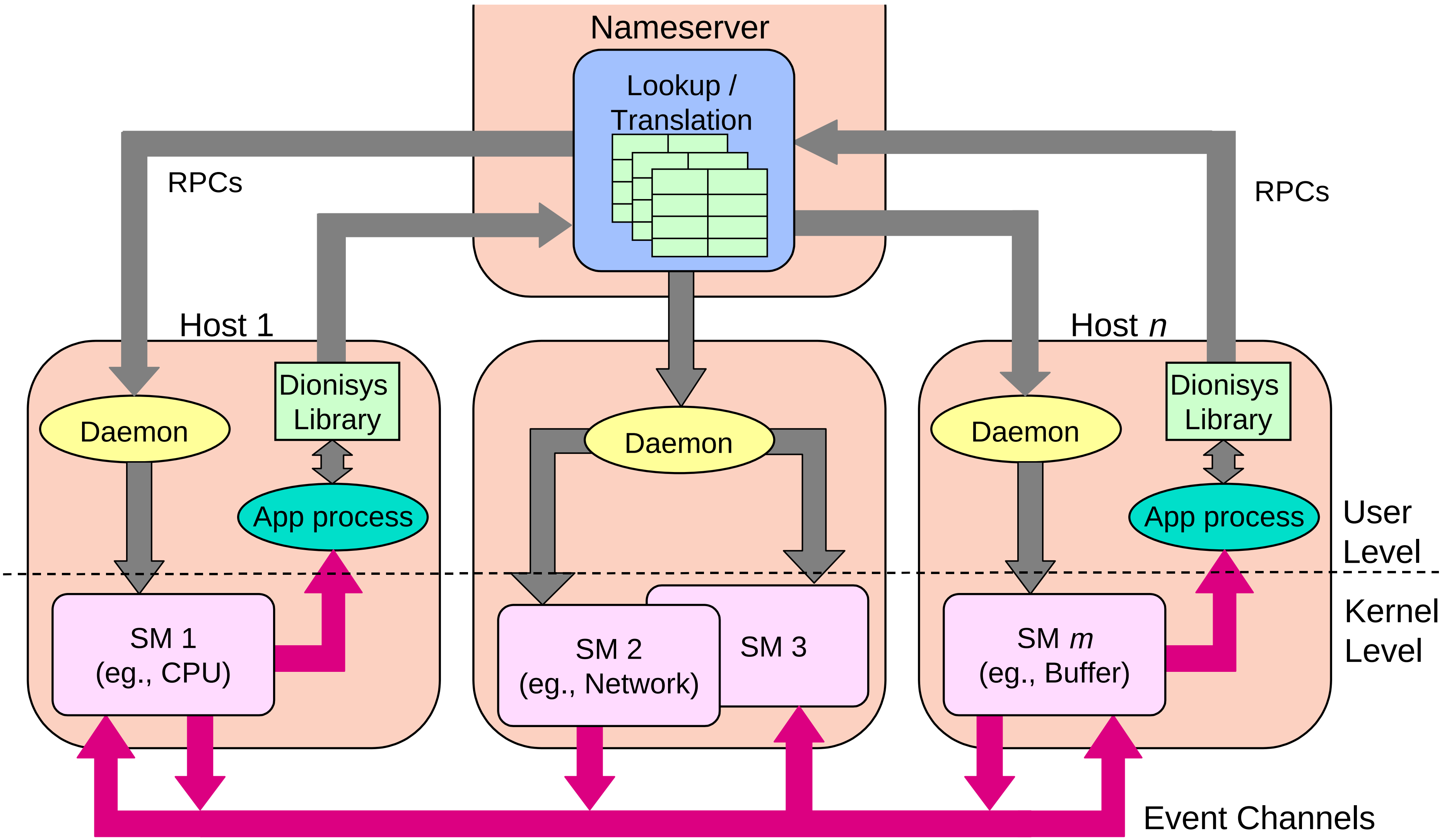
Example System Usage

- Scalable web servers / farms
  - Adaptive load-balancing, caching
- Adaptable protocols
  - For flow, error, rate control etc
- Coordinated resource management
  - e.g., Tradeoffs in CPU versus bandwidth usage

Linux Dionisys Components

- Service extensions:
  - Service managers (SMs)
  - Monitors - influence when to adapt
  - Handlers - influence how to adapt
- MEDEA event channel subsystem
  - Transport events between SMs, where adaptation is needed
- SafeX daemons
- Nameserver, library (API)

Linux Dionisys Overview

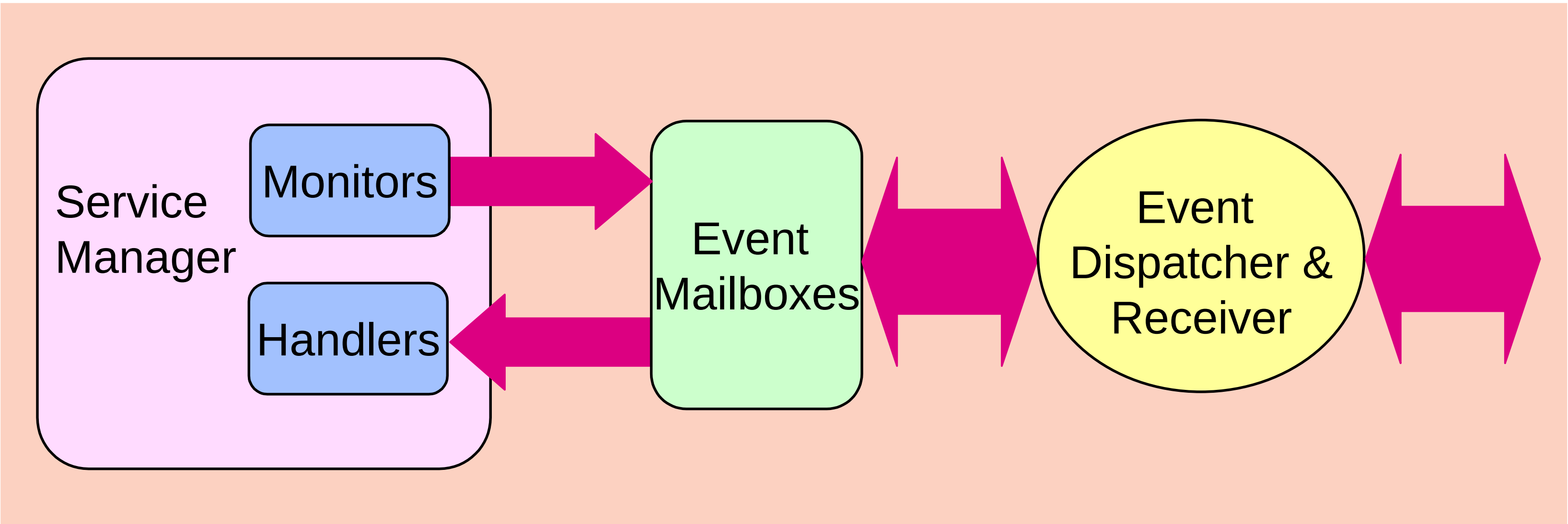


MEDEA

- Provides "event-channels" for communication
  - **One** source (a monitor)
  - Potentially **many** destinations (handlers)
  - Events are asynchronous but may be cascaded
- Provides cross-host, cross-address-space & cross-protection-domain communication
  - e.g., kernel upcalls
- Uses "mailbox" abstractions:
  - One outbox for every monitor
  - One inbox for every service manager

Decoupled Management & Delivery

Computer Science



- MEDEA provides an API for unrestricted event-driven communication

MEDEA Features

- Can batch or select single events for delivery
  - Supports "fast" syscalls that do not block & real-time upcalls
  - Coordinated user-level event delivery and handling
- Prioritized event delivery
  - Can dispatch (receive) events from (into) mailboxes according to an ordering policy
  - Real-time event delivery is possible

SafeX

- Allows app-specific service extensions to be dynamically-linked into kernel address space
  - Can deploy code on remote hosts
- Provides compile- and run-time support to:
  - Enforce bounded execution of extensions
  - Guarantee service isolation (using “guard” fns)
  - Maintain system integrity

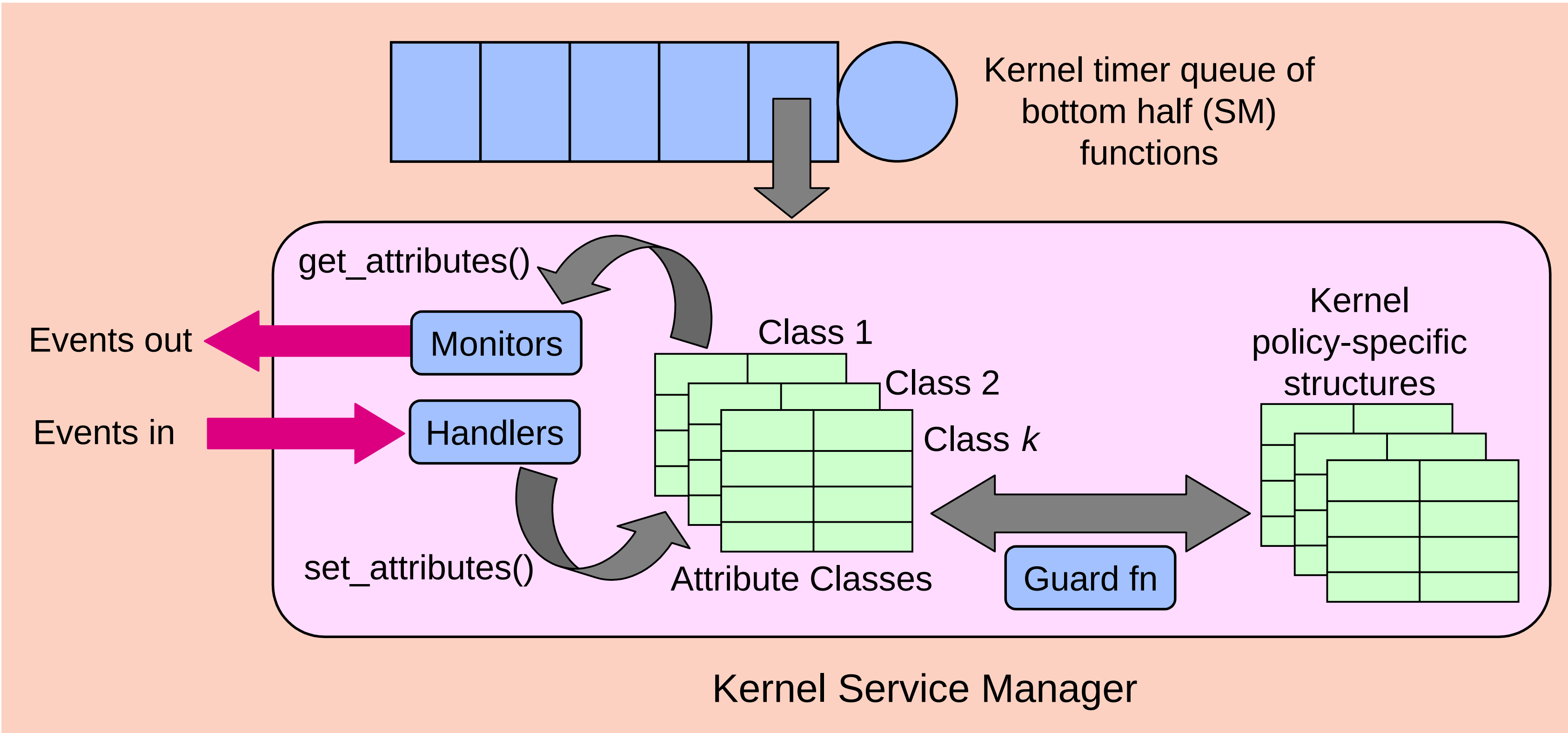
SafeX Features

Computer Science

- Extensions written in Popcorn & compiled into Typed Assembly Language (TAL)
- Memory protection:
  - Prevents forging pointers to arbitrary addresses
  - Prevents de-allocation of memory until safe
- CPU protection:
  - Requires resource reservation for extensions
  - Aborts extensions exceeding reservations
- Interfaces to synchronization objects

A Kernel Service Manager

Computer Science



Experimental Scenario

Computer Science

- CPU service manager monitors CPU utilization and adapts process timeslices
  - Timeslices adjusted by PID function of target & actual CPU usage
  - Monitoring performed every 10mS
- Kernel monitoring functions invoked via timer queue
- User-level approach periodically reads /proc/pid/stat
  - Adapts service via kill() syscalls

Monitors and Handlers


```
void monitor () {
  actual_cpu = get_attribute ("actual_cpu");
  target_cpu = get_attribute ("target_cpu");
  raise_event ("Error", target_cpu - actual_cpu);
}

void handler () {
  e[n] = ev.value; // nth sampled error

  /* Update timeslice adjustment by PID fn of error */
  u[n] = (Kp+Kd+Ki).e[n] - Kd.e[n-1] + u[n-1];

  set_attribute ("timeslice-adjustment", u[n]);
}
```

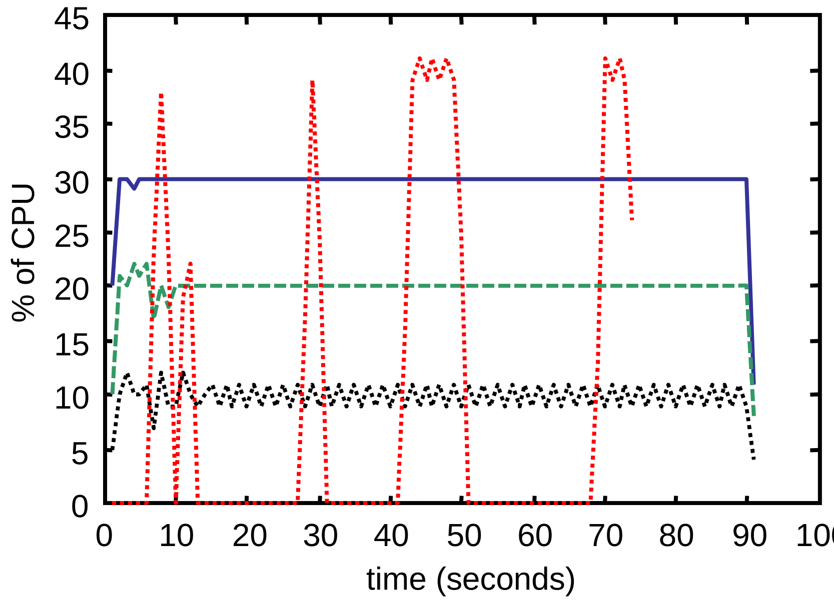
Guard Functions

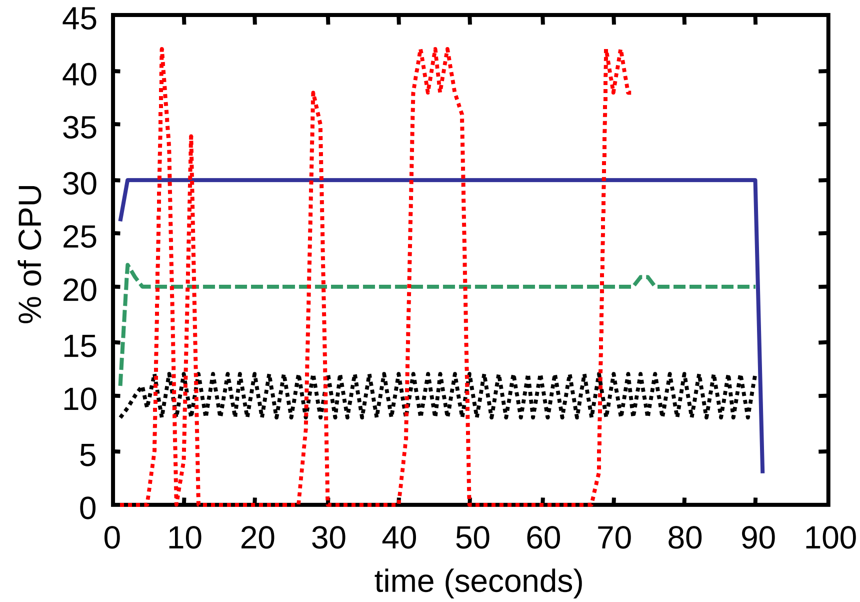
Computer Science

```
// Check the QoS safe updates to a process' timeslice

default_timeslice = target_cpu;

guard (attribute, value):
  if (attribute == "timeslice-adjustment")
    if (value in range [0, 0.25*default_timeslice])
      if (value is QoS safe)
        timeslice = target_cpu + value;
```

CPU Service Management -1

Computer Science

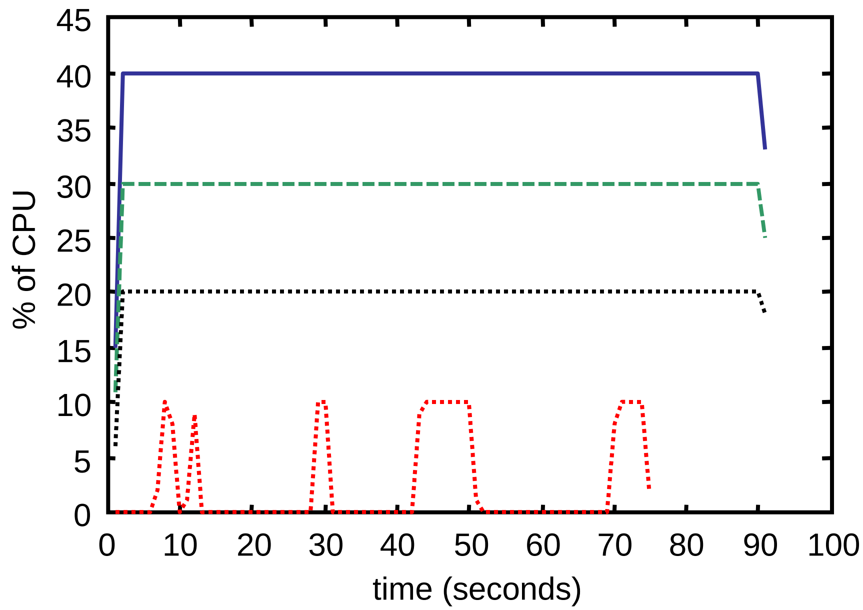


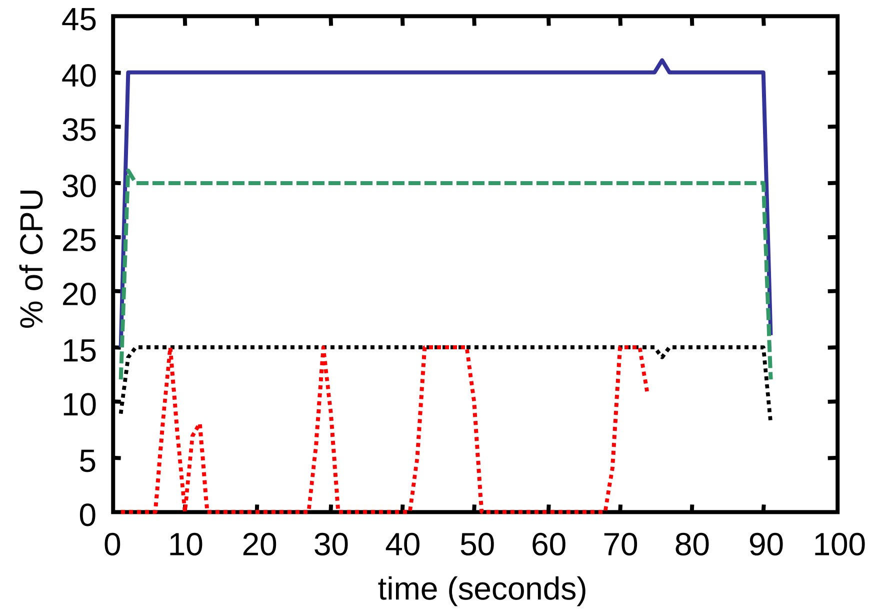


- 3 CPU-bound tasks w/ 30, 20 & 10% target CPU shares
- Less service oscillation in left graph for kernel service management
- Transient overloads do not affect service guarantees

CPU Service Management -2

Computer Science





- 3 MPEG processes with 40, 30 & 20% target CPU shares
- Finer-grained kernel service management is capable of sustaining 20% CPU utilization for 3rd process (left graph)
- User-level management (right graph) cannot meet needs of process with target 20% CPU utilization

Conclusions

Computer Science

- Linux Dionisys supports service extensions to customize system for app-specific needs
- SafeX verifies safety of extensions
  - Extensions may be dynamically-linked into local & remote address spaces
- MEDEA provides event-based communication mechanism that triggers service adaptations
- Overall system improves service to applications