Summer Institute on Software Architecture

Embedded Systems Architecture 2: Real-Time Techniques

Instructor: Calton Pu

calton.pu@cc.gatech.edu

Georgia Tech

Overall Structure (Day 1)

- Introduction to modern embedded systems
  - Ubiquitous computing as a vision for integrating future embedded systems
  - From embedded to resource constrained systems
  - Some basic techniques for constructing real-time embedded system software
- **Principled embedded software infrastructure**
  - **Survey of real-time scheduling algorithms: static, dynamic priority, static priority dynamic**
  - **I/O processing and networking for embedded systems**

Georgia Tech

Static Predictability

- RTES: satisfying the time constraints
  - Certain assumptions about workload and sufficient resource availability
  - Certify at "design time" that all the timing constraints of the application will be met
- For static systems, 100% guarantees can be given at design time
  - Immutable workload and system resources
  - System must be re-certified if anything changes

Georgia Tech

Dynamic Predictability

- Dynamic systems: not statically defined
  - Changeable system configuration
  - Changeable workload
- Dynamic predictability
  - Under appropriate assumptions (sufficient resources)
  - Tasks will satisfy time constraints

Georgia Tech

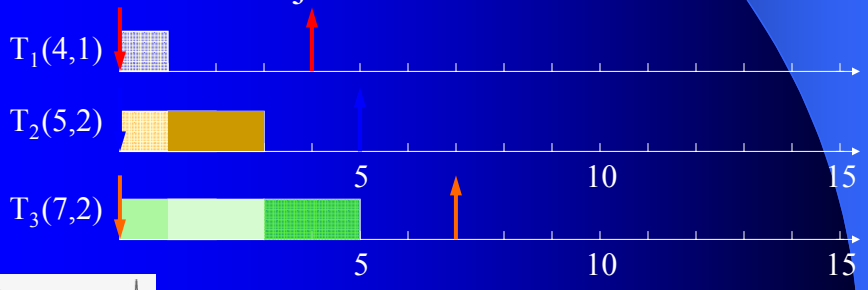
Earliest Deadline First

- Assumptions
  - The schedule is feasible
  - Tasks are independent
- Dynamic priority scheduler
  - Look at all tasks in the queue
  - Compare their deadlines
  - Earliest Deadline First (EDF)
- Minimizes deadline failures when feasible
  - Another story when schedule is infeasible

Georgia Tech

EDF Basics

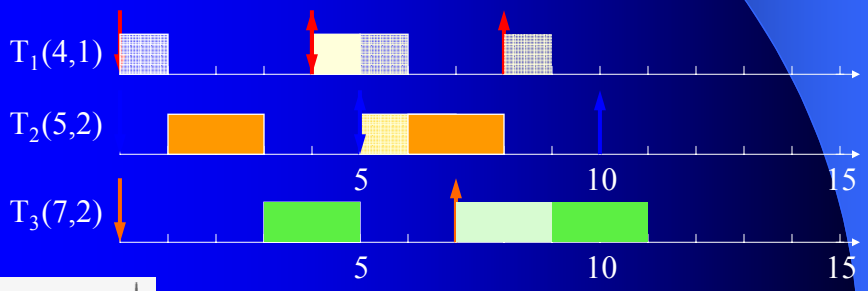
- Optimal dynamic priority scheduling
- A task with a shorter deadline has a higher priority
- Executes a job with the earliest deadline



Georgia Tech

EDF Example

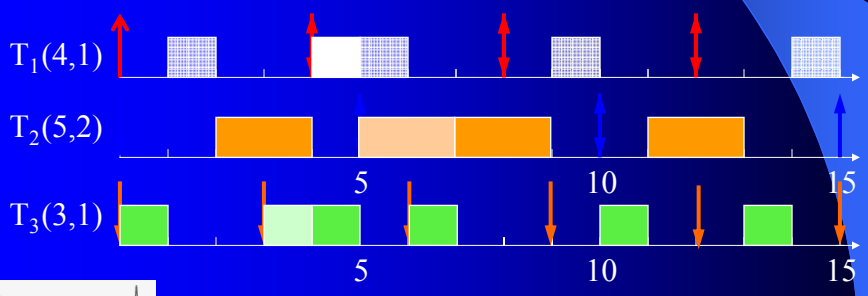
- When the schedule is feasible, EDF can schedule it
- No need for preemption



Georgia Tech

EDF Optimality

- Optimal scheduling algorithm
  - Up to 100% WCAU (Worst Case Achievable Utilization)
- Intuition: can't lose the "most urgent job"



Georgia Tech

Discussion of EDF

- Optimal dynamic scheduler
  - Why do we need anything else?
- Several practical problems
  - Overhead of dynamic scheduling
  - Instability under overload (over the cliff)
  - "Priority inversion"

Georgia Tech

Minimum Laxity First

- Similar to EDF, improved
  - Laxity = time to latest feasible start time (when the task can still complete)
  - Run the task closest to failing first
  - Optimal in minimizing deadline failures
  - More graceful degradation
- Comparison with EDF
  - MLF = EDF when all tasks have same length

Georgia Tech

Discussion

- Limitations of static schedulers
- Limitations of dynamic priority schedulers
- How can we create dynamic schedulers with static priorities?

Georgia Tech

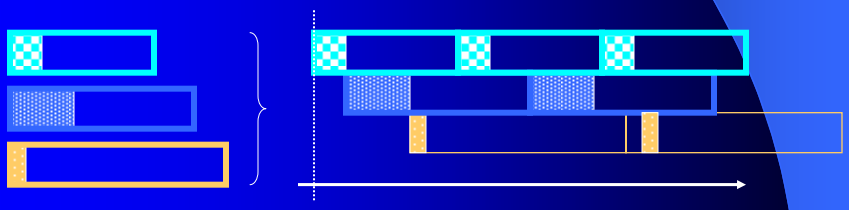
Rate Monotonic Scheduler

- Classic hard real-time CPU scheduler
  - Preemption-based
  - Independent tasks
- Static scheduler: fixed priorities
  - No dynamic priority adjustments
  - Not a static schedule

Georgia Tech

RM Algorithm

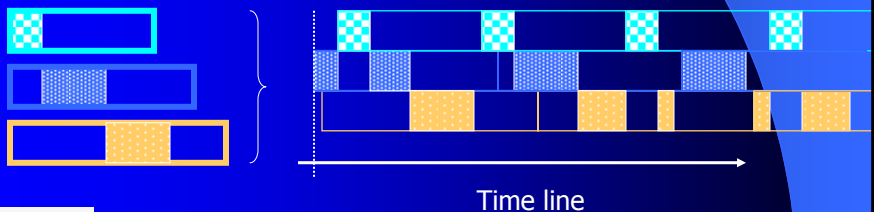
- Main components
  - Periodic tasks, marked by start and end
  - Priority-based preemption
  - Shorter period tasks get higher priority



Georgia Tech

Crucial Zone Theorem

- Only need to check the first period
  - If tasks make their first deadline, then they can make the following deadlines



Georgia Tech

Discussion of RM

- Zhao's evaluation criteria
  - CPU utilization: guaranteed theoretical WCAU limit of 0.693 for CPU
  - Robustness: OK if below the 0.693 limit
  - Timing fault: limited by preemption
  - Aperiodic jobs: high priority until the 0.693 WCAU limit
  - Run-time overhead: may be high

Georgia Tech

Another Discussion

- Sha's list of advantages
  - Fixed priorities: easy management
  - Aperiodic tasks: sporadic server, etc
  - Synchronization: priority ceiling, etc
  - Imprecise computations
  - Ease of implementation

Georgia Tech

Transient Overload

- Variable task CPU consumption
  - Worst case may be too stringent
  - Guarantee a set of critical tasks for the worst case (WCAU)
  - Add other tasks at lower priorities
- During overload
  - Set of highest priority tasks run "normally" (continue to be guaranteed)

Georgia Tech

Period/Priority Mismatch

- Q: high priority task, but long period?
- A: period transformation (task slicing)
  - Divide the task into *k* pieces
  - Run each piece in its own period, size *p/k*
  - Original task completed with the last piece
- Complications and issues
  - Context switch overhead (*k* times)
  - Portability, synchronization

Georgia Tech

Aperiodic Tasks

- Two kinds of aperiodic tasks
  - High priority: emergency alerts
  - Low priority: background jobs
- Problem with background jobs
  - Neglected during transient overload
  - Long response time (example 3, p. 56)

Georgia Tech

High Priority Aperiodic

- Aperiodic server
  - Pretends to be a high priority periodic task
  - Sub-schedules its allotted time slices
  - Can preempt normal periodic tasks
- Constraints and issues
  - Pre-allocation counts towards WCAU limit
  - Independent replenishment and overload handling policies

Georgia Tech

Task Synchronization

- Priority inversion (example 4, p. 57)
  - Priorities combined with locking
  - T3 holding lock(A), is preempted by T1, which needs lock(A)
  - T1 waits for T3, and T3 waits for T2
- Examples of remedies
  - No preemption when holding lock(A) - also introduces priority inversion

Georgia Tech

Priority Ceiling Protocol

- Priority inheritance
  - If T1 waits for T3, then T3 gets T1 priority
- Dynamic rescheduling
  - Critical sections execute at highest priority
  - Waiting is outside critical sections
  - Highest priority waiting task runs next
  - Critical section becomes a subroutine of the highest priority waiting task

Georgia Tech

Advantages of PC

- Deadlock free (example 5, p. 57)
  - Critical section always runs
  - Waiting is outside critical sections
  - T2 prevents T1 from getting lock (fig. 2)
- Bounded priority inversion
  - High priority tasks "jump over" queues
- Good schedulability tests

Georgia Tech

Further Discussion of PC

- Priority inversion
  - Low priority tasks in critical sections become high priority
  - Tasks w/o critical sections may wait
- Execution overhead
  - Every critical section needs a priority queue

Georgia Tech

"Best Effort" Scheduling

- Minimum laxity first (MLF)
  - Laxity = time to latest feasible start time (when the job can still complete)
  - Run the job closest to failing first
  - Optimal in minimizing deadline failures
  - Earliest-Deadline First (EDF) also optimal
  - MLF = EDF when jobs have same length

Georgia Tech

Ada Case Study

- Designed for "safe" programming
  - Predictable program properties
  - Real-time a big application target (AF)
  - One way to do each task
- Some difficulties with real-time
  - Non-deterministic task scheduling
  - Prioritized tasks queued in FIFO order
  - No dynamic change of task priorities

Georgia Tech

"Fixing Ada for RT"

- Adopt priority ceiling protocol (p. 60)
  - Monitor task guards critical section (fig. 4)
  - Priority ceiling emulated by run-time
- Difficulties with Ada specification
  - Monitor task cannot suspend itself (no I/O)
  - Sporadic server for aperiodic tasks
  - Disallow Ada fixed priorities
  - Get around FIFO queues

Georgia Tech

RM: Average Case Analysis

- Stochastic characterization
  - Most likely (average) workload, described by cumulative distribution function (CDF)
  - Utilization determined by scaling workload
- Asymptotic approximation
  - Probabilistic density function of utilization calculated from CDF plus assumptions
  - U depends only on task period, not length

Georgia Tech

Discussion

- Rate monotonic is "state of art" for RTES scheduling
  - Limitations of RM and static schedulers
- Adaptive approaches for unpredictable environments
  - What principles can we apply?

Georgia Tech

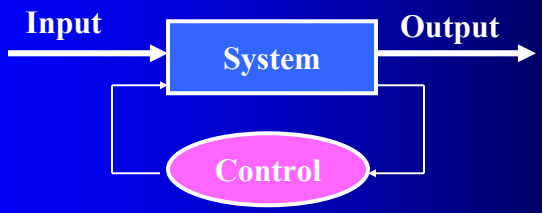
Control Systems

- Systems with changes
  - Human-controlled: bicycle, cars, airplane
  - Inherently unstable airplane wings
  - Chemical reactions, nuclear power plants
- Basic assumptions
  - Changes are within a certain range
  - Changes are "continuous" (physical world)

Georgia Tech

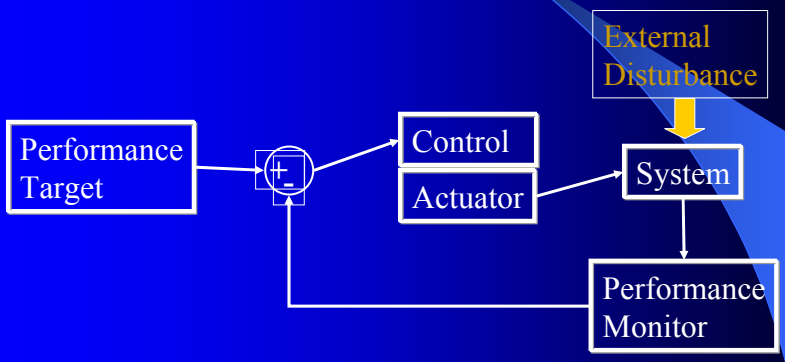
Feedback Approach

- Basic control loop
  - Observe the system being controlled
  - Compare current behavior with expected
  - If different, make adjustments



Georgia Tech

Control Components



Georgia Tech

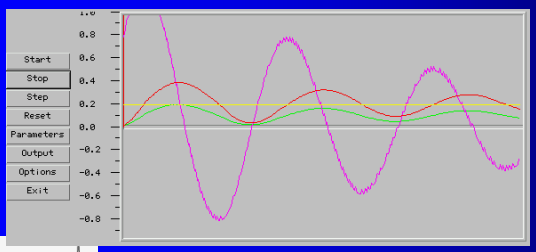
Phase Locked Loop

- Simple example: FM Radio
  - Frequency modulation over a channel
  - Channel is a range of spectrum: 90.1MHz
  - Broadcast frequency moves over range
- Simple loop: PLL
  - Knows about the range, frequency move
  - Moves along with the broadcast

Georgia Tech

Predictability of Control

- Control system properties
  - Stability: does the steady state converge?
  - Maximum error: how far from target?
  - Responsiveness: how fast does it react?



Georgia Tech

PID Control

- Proportional-Integral-Derivative
- Linear combination of 3 components
  - Proportional: C_p
  - Integral: C_i
  - Derivative: C_D
- Lu et al [RTSS'99]

$$Control\ (t) = C_p Error\ (t) + C_I \int Error\ (t)dt + C_D \frac{dError\ (t)}{dt}$$

Georgia Tech

Advantages of PID

- Systematic design method
  - Given predictable input parameters
  - Design control components by composition of filters (Superposition Theorem)
- Predictable performance
  - Responsiveness, maximum error, stability
  - Sampling Theorem: need to sample at twice the frequency of phenomenon

Georgia Tech

PID Design Tradeoffs

- Example: minimize response time
  - React immediately to input change
  - Leads to instability during spikes
- Integral filter adds stability
  - Average the input signals over a window (smooth out the spikes)
  - Reduces instability, delays response

Georgia Tech

FC-EDF

- Goal of the scheduler
  - Maximize utilization & minimize MissRatio
  - EDF maximizes utilization anyway
  - Feedback controls maximum allowed utilization level to minimize MissRatio
- Control knobs (figure 2, p. 59)
  - Admission control: rejection of tasks
  - Service level: multiple versions of tasks

Georgia Tech

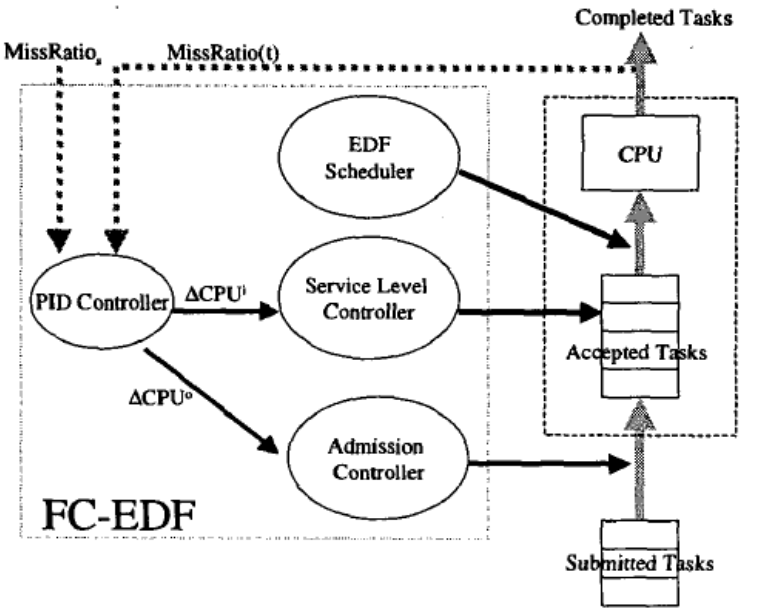


Figure 2 Architecture of FC-EDF

PID Controller

- Observes the MissRatio of system
  - Calculates missed deadlines in a window
  - Sampling rate: every *SP* seconds
- Control signal (ΔCPU) on utilization
  - Calls Service Level for "quick" reaction (small adjustment of accepted tasks)
  - Admission Control called if needed (slower impact on utilization

Service Level Adjustment

- Task service levels
  - Logical versions of tasks that require more or less CPU
  - Concrete example: imprecise computations
- Adjustments that change utilization
  - Higher service level uses more CPU
  - Typical scenario: starts at high level and gets reduced during overload

Georgia Tech

Admission Control

- Stops tasks from entering queue
  - Does not change accepted task queue
  - Reduces CPU utilization slowly
- Can only reduce task queue length
  - Service level is bi-directional control
- Potential trade-off with service level
  - Admit tasks at low level service

Georgia Tech

Discussion

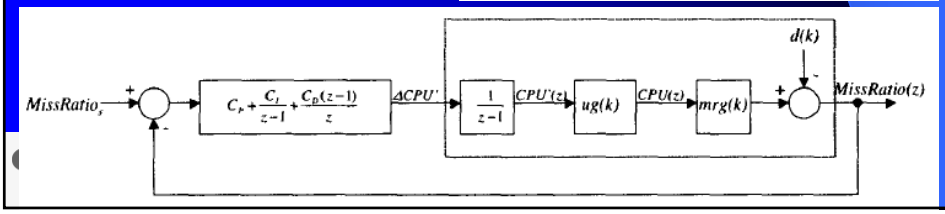
- Disadvantages
  - Coarse-grained adaptation (seconds)
  - Detects missed deadlines and then react
  - Applicable to stable workloads
- Advantages
  - Close enough to classic feedback model for a classic analysis (section 3.6)

Georgia Tech

Stability Analysis

- BIBO stability
  - Bounded Input → Bounded Output
- Assumptions
  - Utilization tracking is good (bounded error)
  - Tasks are independent
- Stability condition

1. $C_I > 0$, $|C_D| < 1$, $2C_P - C_I + 4C_D < 4$, and $2 - 2C_D^2 > C_D C_P + C_P - C_I > 0$
2. $C_I = 0$, $|C_D| < 1$, and $0 < C_P + 2C_D < 2$



Other System Parameters

- Maximum error
  - How far can the utilization tracking drift?
- Responsiveness
  - How quickly will the controller respond to input changes?
- Tradeoffs between parameters

Performance Evaluation

- Simulation-based (figures on p. 63)
  - MRA: miss ratio among admitted tasks
  - UTIL: actual CPU utilization
  - HRS: hit ratio among submitted tasks (successful execution ratio)
  - VCR: value completion ratio (higher service level contributes higher value)
  - etf: estimated time factor

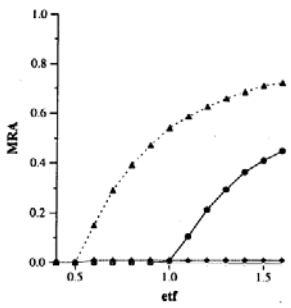


Figure 5 MRA vs. etf

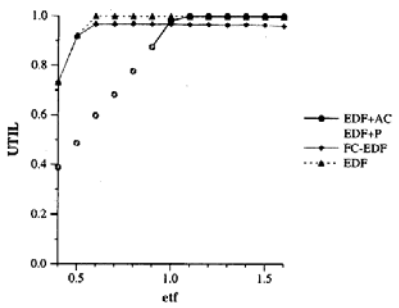


Figure 6 UTIL vs. etf

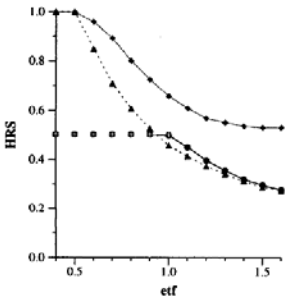


Figure 7 HRS vs. etf

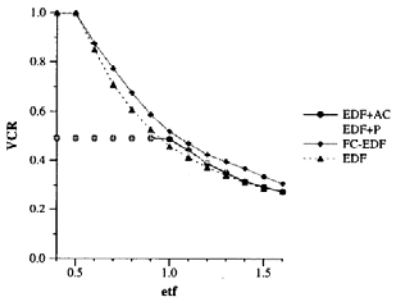


Figure 8 VCR vs. etf

Scheduler Adaptation

- FC-EDF (figure 9, p. 65)
  - Watch the adaptation of FC-EDF
  - etf spike (300 SP) causes misses
  - etf loss (600 SP) causes low utilization

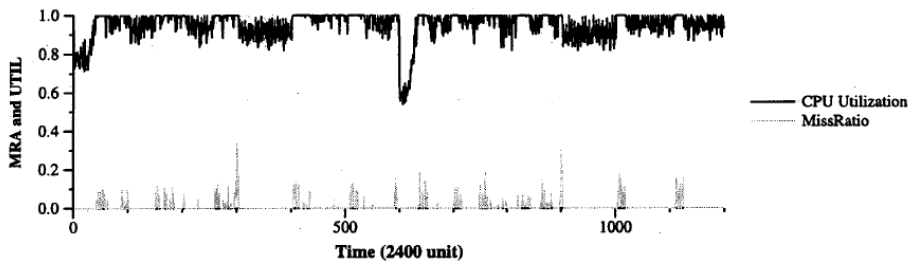


Figure 9 Sampling MissRatio and CPU Utilization of FC-EDF

Scheduler Adaptation

- EDF and EDF-AC
  - EDF (figure 11): high miss ratio
  - EDF-AC (figure 10): improves on EDF

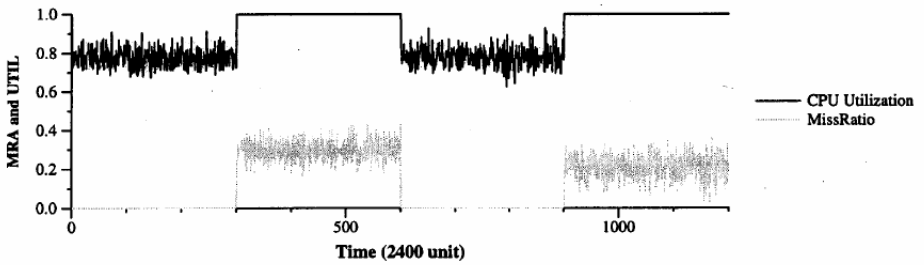


Figure 10 Sampling MissRatio and CPU Utilization of EDF+AC

Scheduler Adaptation

- EDF and EDF-AC
  - EDF (figure 11): high miss ratio
  - EDF-AC (figure 10): improves on EDF

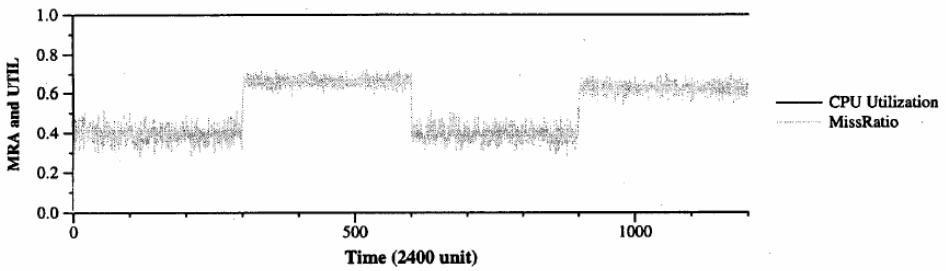


Figure 11 Sampling MissRatio and CPU Utilization of EDF

Discussion

- EDF, EDF-AC
  - Earliest Deadline First (naïve)
  - EDF with admission control
- FC-EDF
  - Feedback control of service level & AC
  - Stability analysis
  - Performance evaluation
- Problem: miss ratio not really suitable

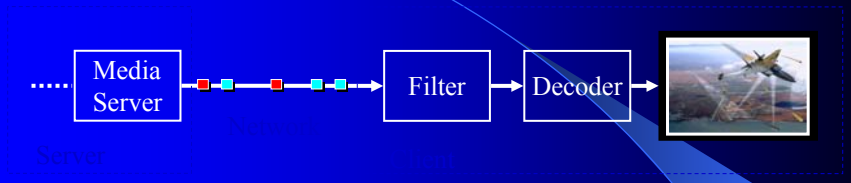
Georgia Tech

Feedback-Based Scheduling

- Problems with using miss ratio as feedback metric
- Feedback should be fine-grained
- How do we use feedback directly in scheduling?
- How do we "automate" the feedback?
- Steere et al [OSDI'99]

Georgia Tech

Motivation



- Real-rate applications
  - Real-world performance demands
- Automated rate matching
  - fine-grain adjustment of allocation
  - dynamic response to variable rates
  - low programming complexity

Georgia Tech

Priority Schedulers

- Give CPU to highest priority task
- Scheduling policy
  - Assigns priorities to tasks
- Scheduling mechanism
  - Sort runnable task queue according to priorities
  - Run the task at the head of the queue
- Example of problems
  - Priority inversion

Georgia Tech

Proportional Scheduler

- Give tasks a proportional share of CPU
- Scheduling policy
  - Assign proportions to tasks
- Scheduling mechanism
  - Round-robin queue
  - Time slice allocated according to the proportion
- Example of problems
  - Need to cycle through queue fast

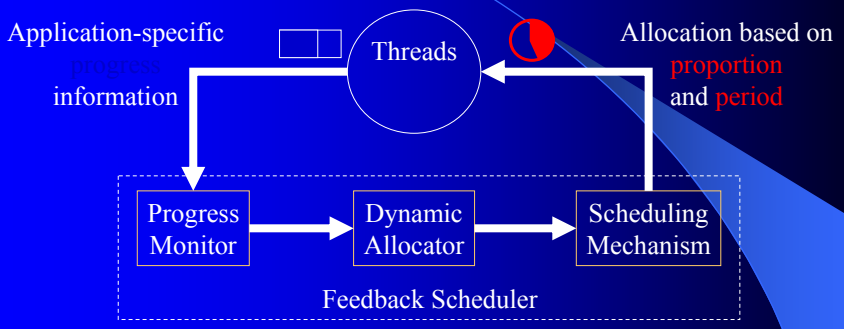
Georgia Tech

Current Schedulers

- Priority-based schedulers
  - all-or-nothing or equal share: coarse grained
    - unless aided by the application programmer: high complexity
- Proportional-share schedulers
  - require program to specify resource needs: high complexity
  - hard to know for variable rate apps: lack dynamic response

Georgia Tech

Feedback Approach



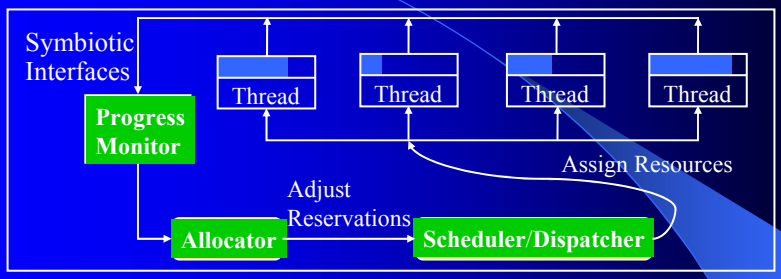
Georgia Tech

Simple Idea

- Real-rate task model
  - Known CPU requirements
  - Known deadline
- Dynamic scheduler
  - Monitors task progress towards deadline
  - Insufficient CPU: increase allocation
  - Do it fast enough before miss

Georgia Tech

Our Approach



- Application progress monitor
- Resource reservation scheduler
- Dynamic allocator:
  - Feedback based allocator samples progress
  - Calculates resource needs, assigns allocation

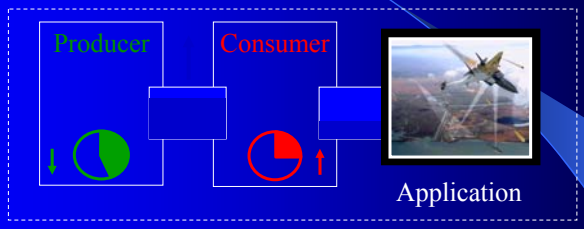
Georgia Tech

Progress Estimation

- Automated monitoring feasible
  - Symbiotic interfaces between system and application
- Concrete examples
  - Server: consumer of a bounded buffer
  - Data rate of shared memory queues, sockets, pipes, etc
  - I/O intensive: consumers of the I/O subsystem
  - Interactive: listen to tty instead of sockets

Georgia Tech

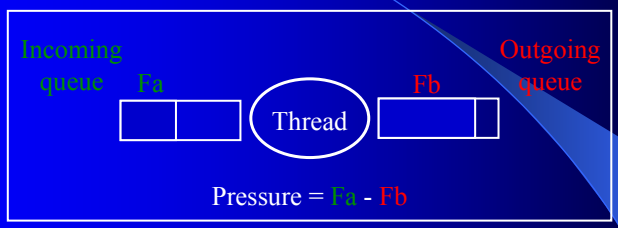
Queue Example



Monitor Progress

Feedback Scheduler → Assign Allocation to Producer, Consumer

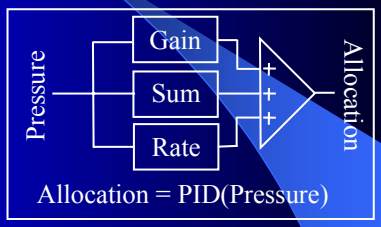
Progress Monitor



- Fill levels show data flow pressure
  - High pressure: increase CPU allocation/priority
  - Low pressure: decrease allocation/priority
  - Goal: keep data flowing smoothly

Dynamic Allocator

- Real-rate: pressure fed to PID controller
- Real-time: progress towards reservation fed to PID controller
- Other: constant positive pressure (bypass adaptation)



Georgia Tech

Overload Allocation Policy

- Real-time:
  - renegotiate reservations, admission control
- Others:
  - weighted fair sharing, proportional squishing
  - Transient load: jobs regain allocation
  - Long-term load: quality exceptions to applications
    - Applications can shed load
    - Increase job importance

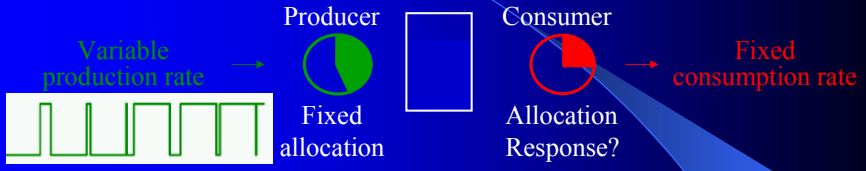
Georgia Tech

Scheduling Mechanism

- Reservation scheduler
  - based on proportion and period
  - uses rate-monotonic scheduling
  - allocations enforced during process dispatch
  - allows fine granularity allocation adjustment
  - low overhead for changing reservations
  - respects reservations for applications that specify them

Georgia Tech

Allocator's Response

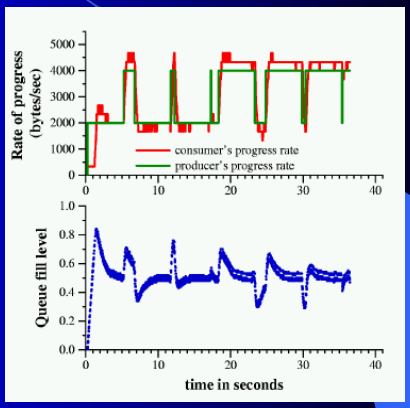


- Response to varying progress
- Vary the progress rate of the producer
  - fixed allocation, variable production rate
- Measure the progress rate of the consumer
  - variable allocation, fixed consumption rate
- Progress rate = allocation X (production/consumption rate)

Georgia Tech

Results on Idle System

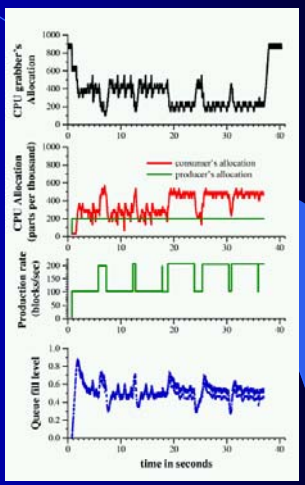
- Allocator response
  - adjusts consumer's allocation rapidly
  - matches producer's progress rate



Georgia Tech

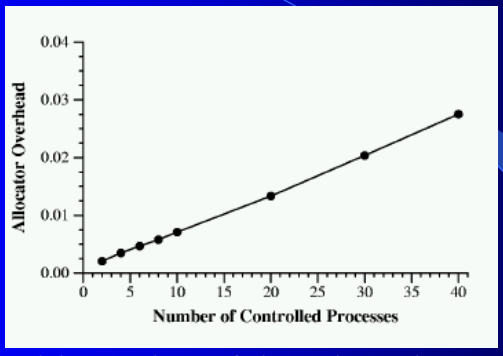
Results on Loaded System

- The "other" load does not have a progress metric
- "Other" load loses allocation to the real-rate consumer job



Georgia Tech

Allocator Overhead



- Linear with number of threads under our control
- Small slope of prototype - 0.06% per process

Georgia Tech

Benefits of Our Approach

- Finer grained allocation
- Avoids starvation and priority inversion
- Easy admission control
- Automatic estimation of reservations
- Implementation in Linux
  source code available at
  http://www.cse.ogi.edu/DISC/projects/quasar/releases

Georgia Tech

Related Work

- Real-time priorities
- Proportional allocation [Waldspurger]
- Reservation Schedulers [Nieh, Jones]

  Previous work focuses on satisfying requirements, rather than inferring requirements accurately
- Balancing between real-time and normal jobs
- Pipeline based allocation [Jeffay]

Georgia Tech

Adaptive Controller

Proportion Specified	Progress Metric	Period Specified	Period Unspecified
Yes	N/A	Real-time	Aperiodic Real-time
No	Yes	Real-Rate	
	No	Miscellaneous	

- Real-time: controller typically doesn't touch specification
- Aperiodic Real-time: controller assigns default period
- Real-Rate: controller uses progress metric to estimate pressure
- Miscellaneous: heuristic, constant pressure

Georgia Tech

Summary

- Unified scheduling mechanism
  - real-rate, traditional and real-time jobs
  - expose application-level progress
- Move scheduling to a higher level of abstraction
  - think progress rather than allocation
- Dual strategy
  - proportional share scheduler: gear box
  - dynamic allocator: automatic transmission

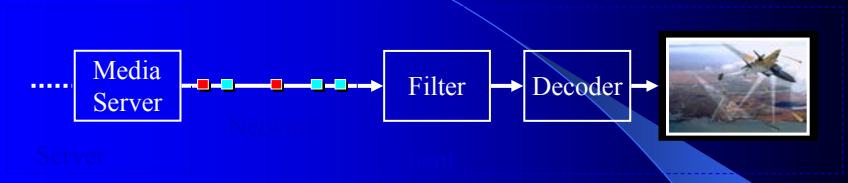
Georgia Tech

Discussion

- Network bandwidth scheduling: Allocate network bandwidth among competing tasks
- Problems with traditional queue schedulers (e.g., FIFO)
- Problems with adapting traditional CPU schedulers for networking (e.g., priority)
- Li et al [MMCN'01]

Georgia Tech

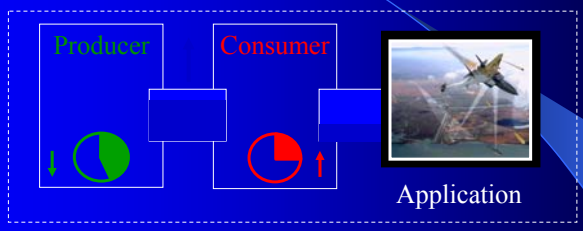
FB-Based CPU Scheduling



- Real-rate applications
  - Real-world performance demands
- Automated rate matching
  - fine-grain adjustment of allocation
  - dynamic response to variable rates
  - low programming complexity

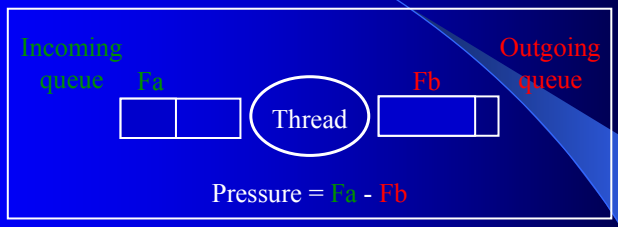
Georgia Tech

Queue Example



Georgia Tech

Progress Monitor



- Fill levels show data flow pressure
  - High pressure: increase CPU allocation/priority
  - Low pressure: decrease allocation/priority
  - Goal: keep data flowing smoothly

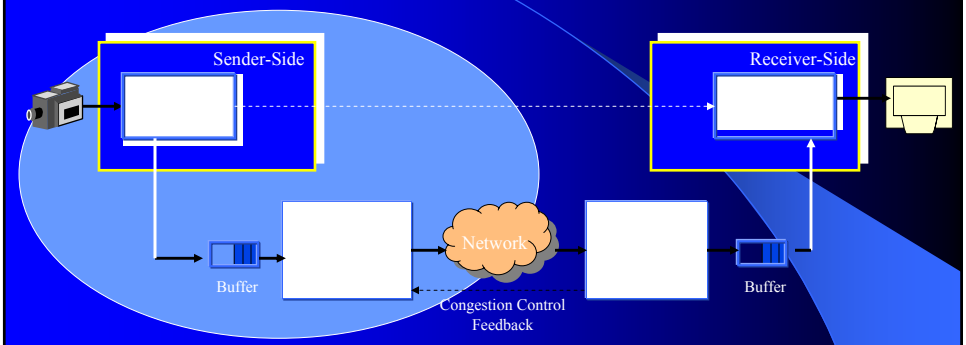
Georgia Tech

Real-Rate Applications

- Real-Rate applications
  - IP telephony
  - Videoconferencing
  - Remote sensors over network
- Common aspects:
  - *Delay-sensitive*
    - data must be delivered from sender to receiver in bounded time.
  - *Self-rate-regulated*
    - e.g. live video source from camera
    - break our infinite source assumption

Georgia Tech

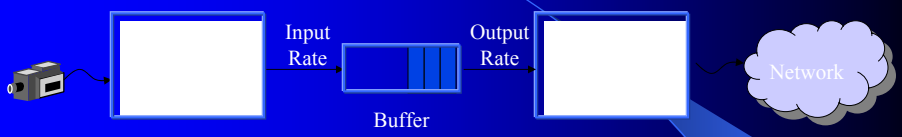
Real-Rate Applications in the Internet



- A real-rate application using a congestion control protocol.

Georgia Tech

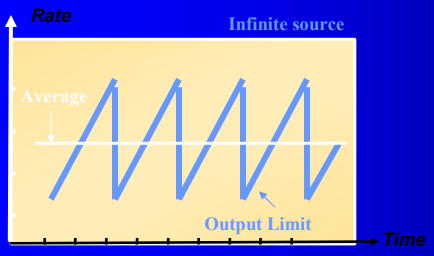
Real-Rate Application

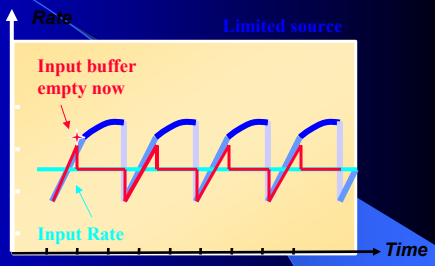


- Buffering delay between real-rate application & congestion control protocol:
  - It is caused by rate mismatches,
    - Input rate is controlled by the application adaptation.
    - Output rate is controlled by the congestion control.
- TCP-friendly congestion control

Georgia Tech

TCP with Finite Source Rate



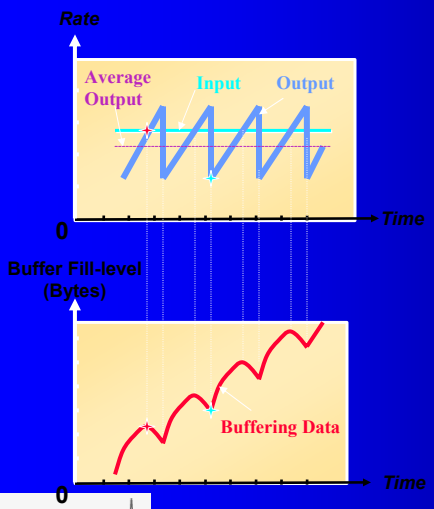


- Infinite sources
  - TCP expands its window at "full speed".

$$\text{Average TCP Throughput} = \frac{C}{RTT * \sqrt{p}}$$

- Finite sources
  - TCP is in "full speed" when input buffer has data.
  - Less aggressive when input buffer is empty.
- Can't model TCP independently!

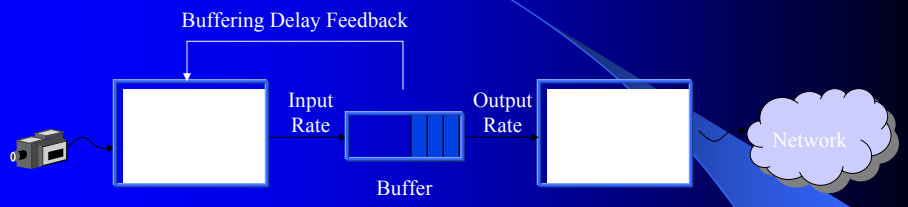
If Input Rate is Higher Than Output

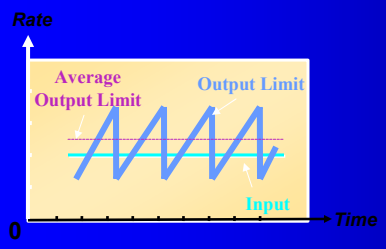


- Assumptions:
  - CBR input
  - AIMD saw-tooth output
  - Input rate is higher than the average of the output Rate.
- Result:
  - Buffering delay builds up
- Solution:

  Application Adaptations

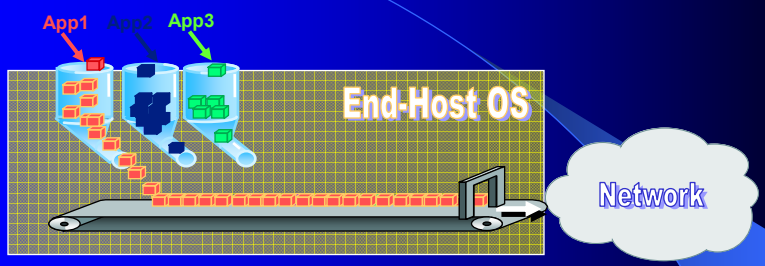
QoS Adaptation Based on Local Buffer





- Adaptation: Reducing input rate to be less than the average of output.
- System goes to the finite source case!

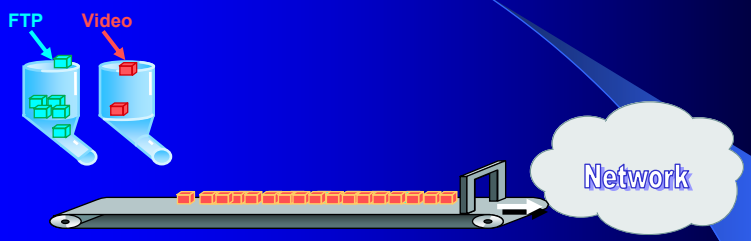
Packet Scheduler



Manages outgoing network interface bandwidth
- Buffers & multiplexes different streams
- Determines *inter-packet intervals* and overall *bandwidth allocation* for each stream

Conventional End-Host OS uses FCFS policy

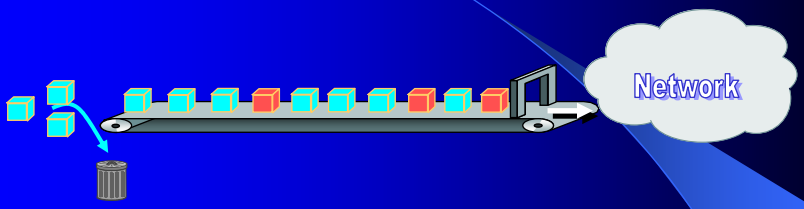
FCFS Packet Scheduling Problems



- Unpredictable delays
  - Many FTP packets may be inserted between consecutive video packets
  - FCFS scheduling does not guarantee service intervals

Georgia Tech

FCFS Packet Scheduling Problems



Competition for bandwidth

Best-effort streams are *greedy*

Real-rate streams are *self-controlled*

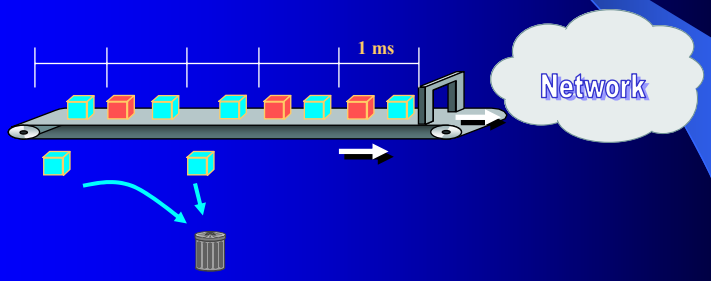
Real-rate packets are dropped regardless of available bandwidth

FCFS scheduling does not guarantee bandwidth

Georgia Tech

Bandwidth Reservation Mechanism

- Each stream is guaranteed a bandwidth *proportion (%)* over some *period (ms)*



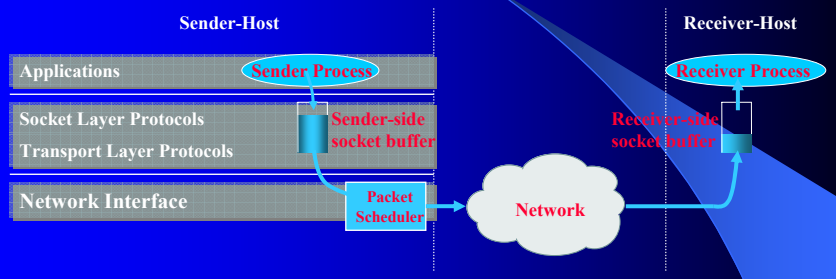
Georgia Tech

Specifying Reservations

- Informed
  - Too complex for programmers or users to specify
  - Bandwidth requirements are not constant
- Inferred
  - Current schedulers use only resource-level information
- Our Solution
  - Packet scheduler dynamically infers application requirements from protocol state

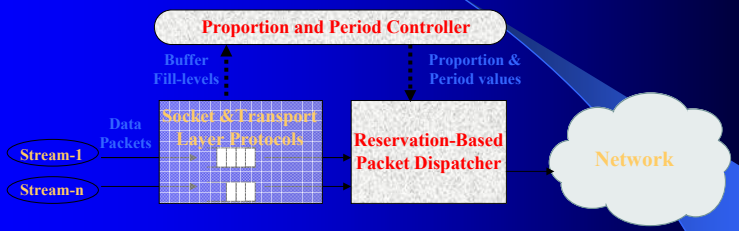
Georgia Tech

Inferring Requirements



- Monitor sender and receiver *socket buffer fill-levels*
- Receiver's socket buffer is on a remote host!
- Data rate requirements change dynamically

Packet Scheduler Architecture



- Packet dispatcher - *reservation-based mechanism*
- Proportion and period controller - *progress-driven policy*

RAMP in Kernel

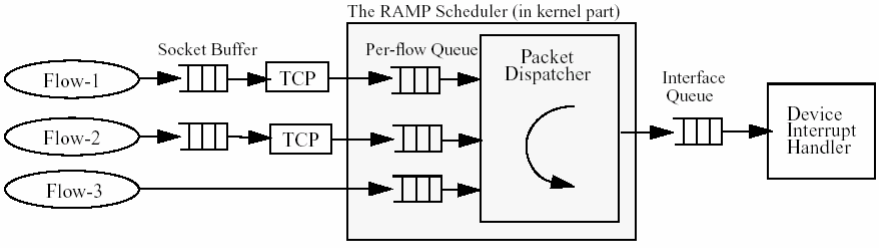


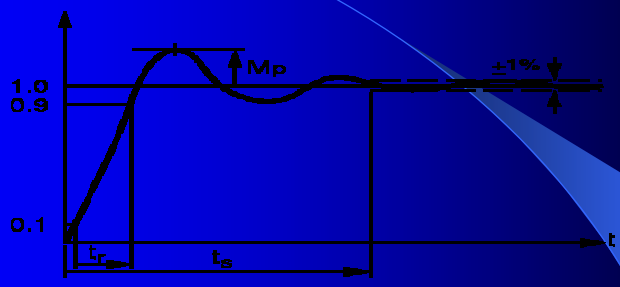
Figure 3: The Packet Dispatcher in an End-host's OS

System Overhead

Maximum Interface Throughput

	TCP Stream Maximum Throughput	UDP Unidirectional Stream Maximum Throughput	CPU Utilization for full TCP throughput
Linux 2.0.35	7.41 Mbits/s	9.41 Mbits/s	2.5%
Linux with our Packet Scheduler	7.41 Mbits/s	9.41 Mbits/s	2.93%

System Responsiveness



- Rise time (tr) - how fast the controller response
- Overshoot (mp) - how much oscillation the output has
- Settling time (ts) - when the result will reach stable

RAMP Response

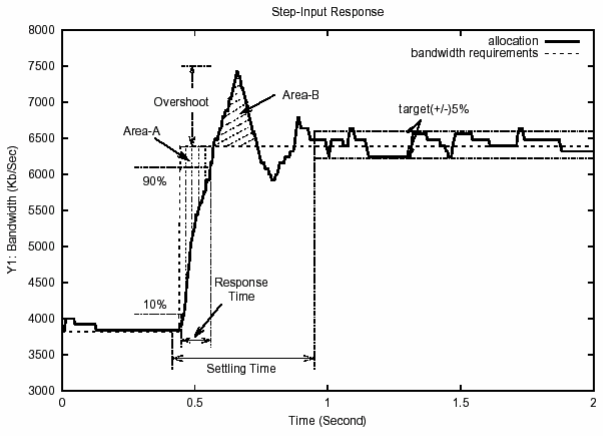
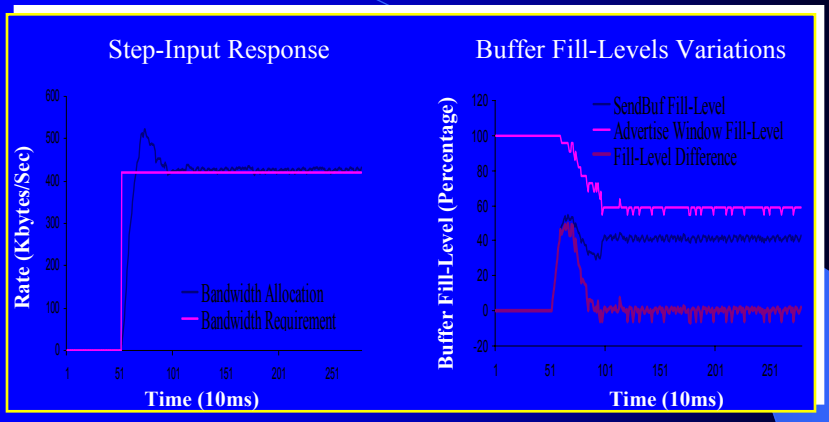


Figure 6: RAMP scheduler's step-input response

System Responsiveness Results

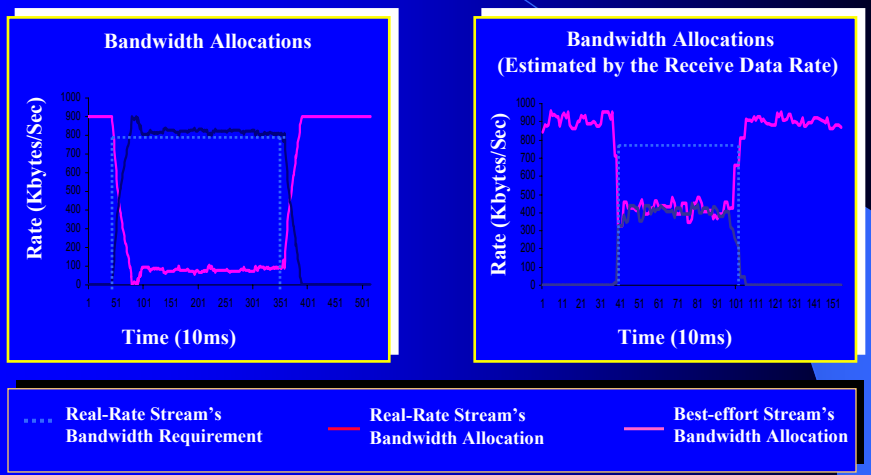


- ***Rise time*** 100ms, ***Overshoot*** 25% and ***Settling Time*** 470ms .
- Controller's setting: PID parameters (4.0, 0.4, 1.0) and Sampling time (10ms) .

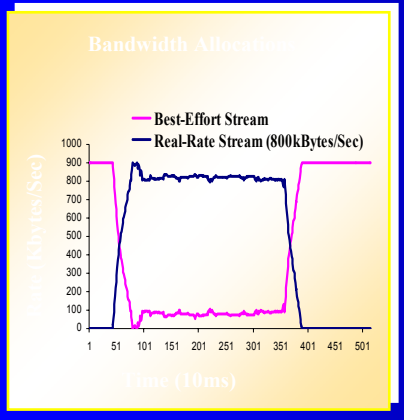
Competition for Bandwidth

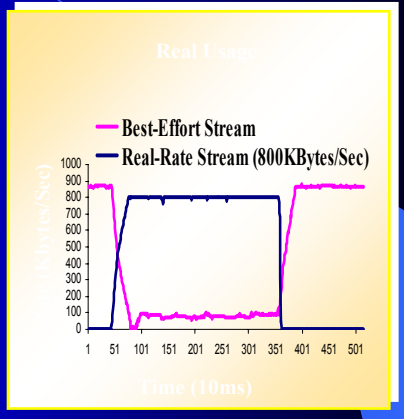
Progress-driven Packet Scheduler

Linux Packet Scheduler



Allocation versus Usage





RAMP Allocation

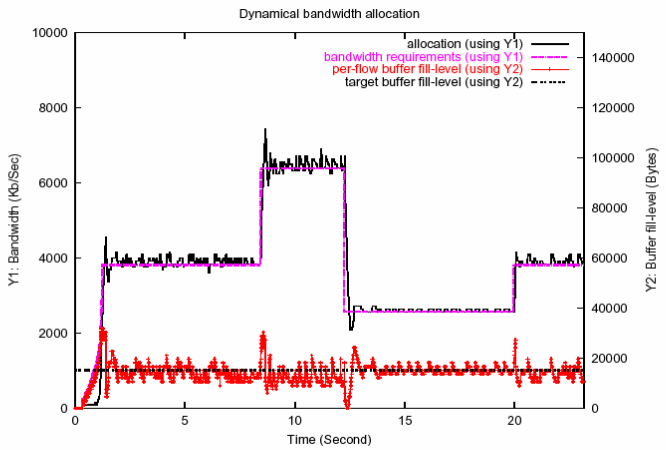


Figure 5: RAMP scheduler's allocations versus the benchmark application's requirements variations

FCFS Scheduler (TCP)

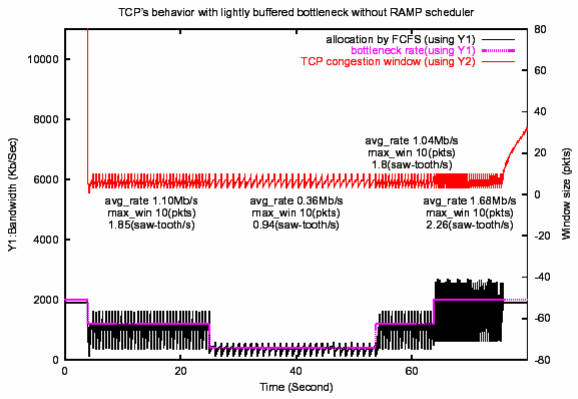


Figure 7: TCP's behavior with a FCFS scheduler

RAMP Scheduler (TCP)

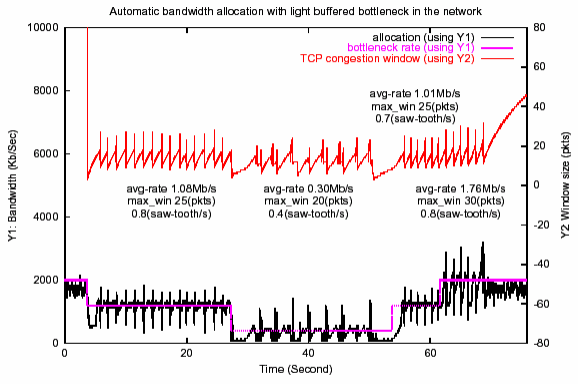


Figure 8: TCP's behavior with a RAMP scheduler

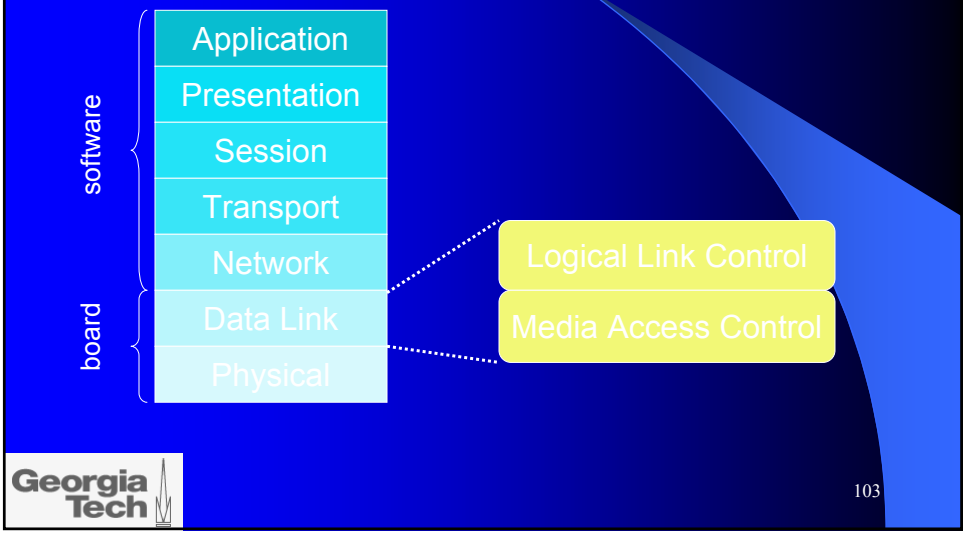
Summary

- Feedback control can be used for several system resources:
  - CPU scheduling (proportional share)
  - Network bandwidth scheduling (rate matching)

Discussion

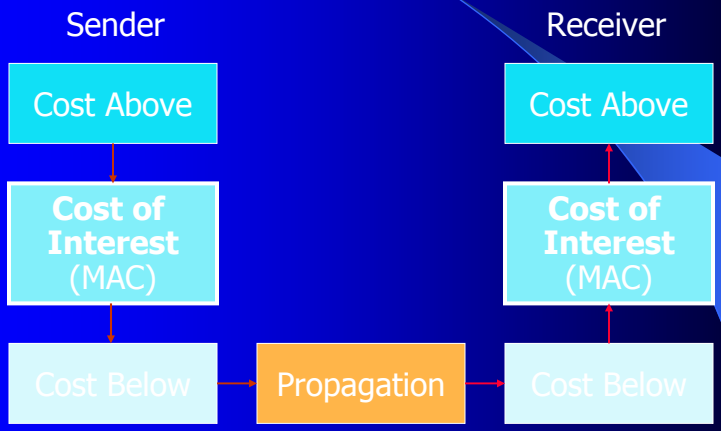
- Application of rate monotonic to network bandwidth scheduling

OSI Reference Model



Georgia Tech

Transmission Time



Georgia Tech

Message Classification

- Synchronous messages
  - Periodic, predictable stream (sensor data)
  - Arrival time: job creation
  - Length: job resource requirement
  - Deadline: job completion requirement
- Asynchronous messages
  - Aperiodic task communications, alerts
  - Unpredictable arrivals, length, deadline

Georgia Tech

Access Arbitration

- Network as a shared resource
  - Rate monotonic scheduling of bandwidth
- Desirable properties
  - Bandwidth utilization (for synchronous)
  - Robustness: not brittle
  - Timing fault isolation and limitation
  - Accommodate asynchronous messages
  - Low run-time overhead

Georgia Tech

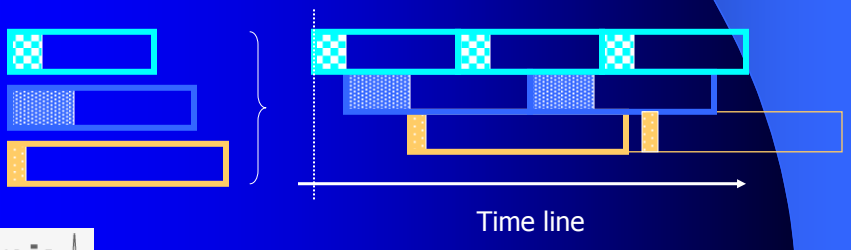
Rate Monotonic Scheduler

- Classic hard real-time CPU scheduler
- Decides who gets the resource next
  - CPU in the original work
  - Network bandwidth in MAC
- Static scheduler: fixed priorities
  - No dynamic priority adjustments
  - Not a static schedule

Georgia Tech

RM Scheduling Algorithm

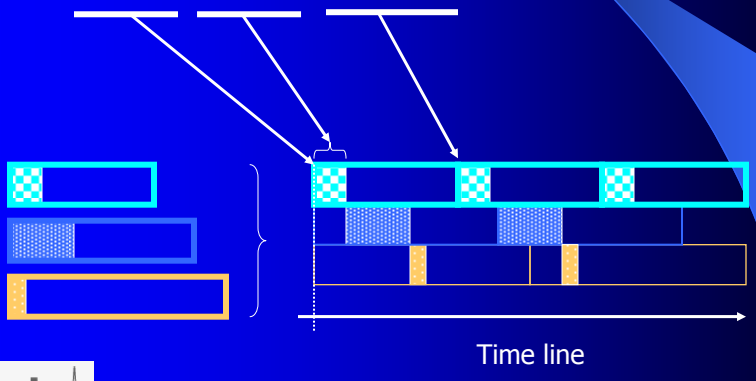
- Assumptions
  - Jobs are periodic, marked by start and end
  - Jobs are independent
- Shorter period jobs get higher priority



Georgia Tech

RM for Networking

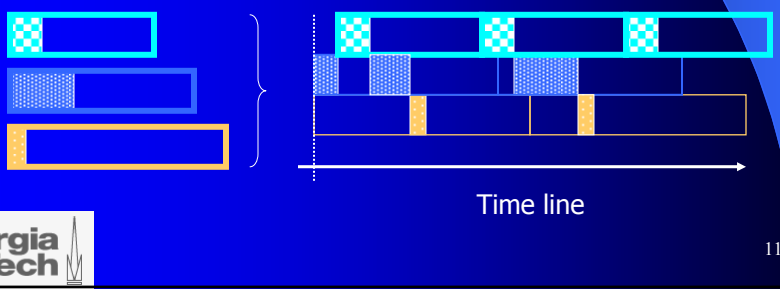
- Synchronous messages
  - Arrival, length, deadline



Georgia Tech

Preemption in Networks

- Simulate preemption
  - Divide message into packets (unit of scheduling)



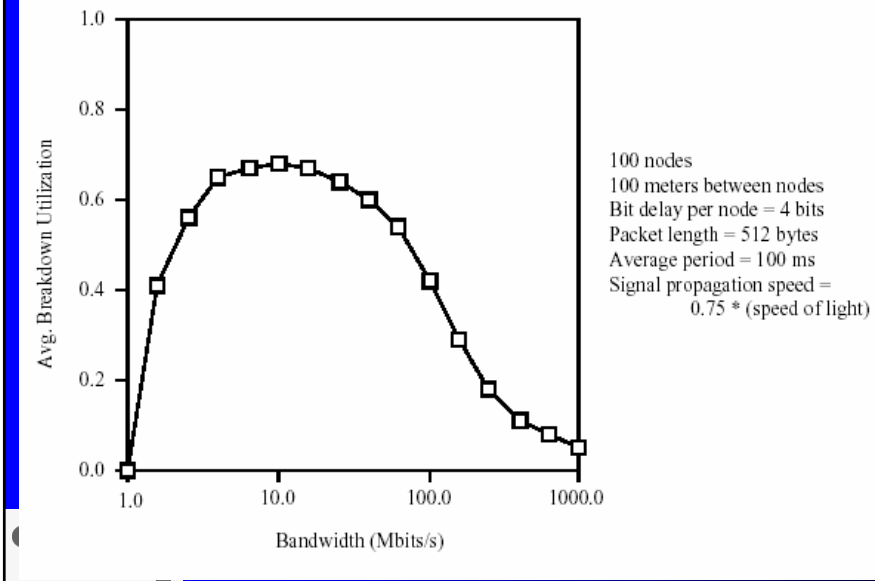
Georgia Tech

Discussion of RM

- Evaluation criteria
  - Bandwidth utilization: guaranteed theoretical limit of 0.693 for CPU
  - Robustness: OK if below the 0.693 limit
  - Timing fault: limited by preemption
  - Asynchronous messages: high priority until the 0.693 limit
  - Run-time overhead: may be high (fig. 4)

Georgia Tech

Priority-Driven Protocol



Transmission Control

- Timed token protocol
  - Nodes are arranged in a ring
  - Each node receives the token in sequence
  - Node k transmits H_k packets (predefined)
  - Token rotation time bounded (half of minimum deadline)
  - If token arrives early, send asynchronous messages

Georgia Tech

Scheduling Analog

- Timed token protocol
  - Nodes are arranged in a ring (round robin)
  - Each node receives the token in sequence
  - Node k transmits H_k packets (time slice)
  - Token rotation time bounded (to guarantee the equivalent of preemption)
  - If token arrives early, send asynchronous messages (different priorities)

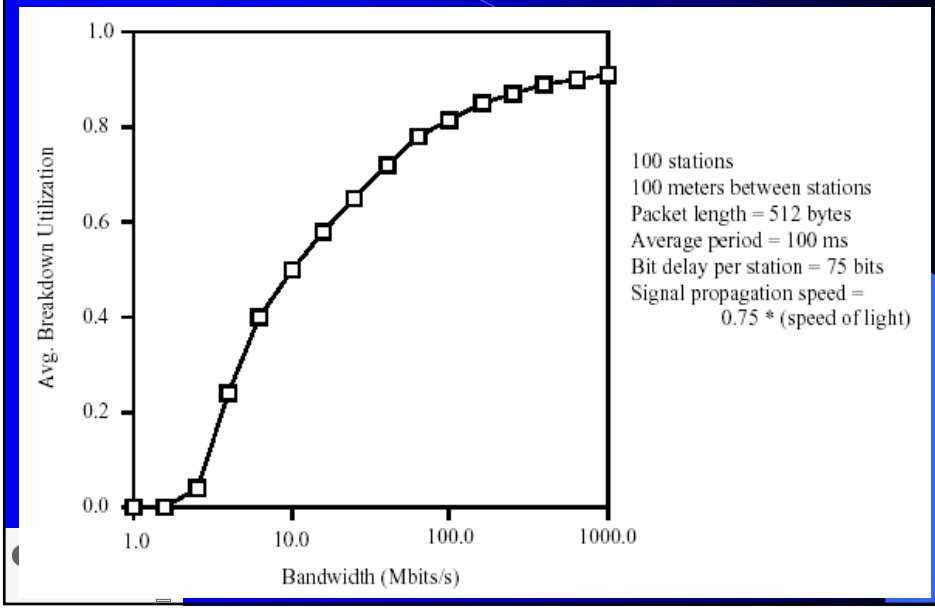
Georgia Tech

Discussion of Timed Token

- Evaluation criteria
  - Bandwidth utilization: 0.33 WCAU
  - Robustness: OK if below the 0.33 limit (Shortest Path), and 0.45 (heuristic)
  - Timing fault: limited by H_k (time slice)
  - Asynchronous messages: squeezed in
  - Low run-time overhead: good scalability (fig. 5)

Georgia Tech

Timed Token Protocol



Asynchronous Messages

- Unpredictable arrivals
- Guarantees in dynamic scheduling
  - **Periodic server**: reserved bandwidth (affects synchronous messages)
  - **Conservative estimation**: worst case analysis of all messages (under-utilization)
  - **Dynamic reservations**: control message to reserve future bandwidth (overhead)

Georgia Tech

"Best Effort" Scheduling

- Minimum laxity first (MLF)
  - Laxity = time to latest feasible start time (when the job can still complete)
  - Run the job closest to failing first
  - Optimal in minimizing deadline failures
  - Earliest-Deadline First (EDF) also optimal
  - MLF = EDF when jobs have same length
  - *"deadline-driven scheduling"*

Georgia Tech

MLF in Networking

- MLF in bandwidth scheduling
  - Send first the messages w/ minimum laxity
- Some drawbacks when priority-based
  - Overhead in priority arbitration
  - MLF requires dynamic re-prioritization
  - Priority inversion due to insufficient number of priority levels

Georgia Tech

Summary of Survey

- Message (job) classification
  - Synchronous messages (periodic jobs)
  - Asynchronous messages (aperiodic jobs)
- Scheduling strategies
  - Access arbitration: who gets to use the resource (network bandwidth or CPU)
  - Transmission control: how long one gets to use the resource (bandwidth)

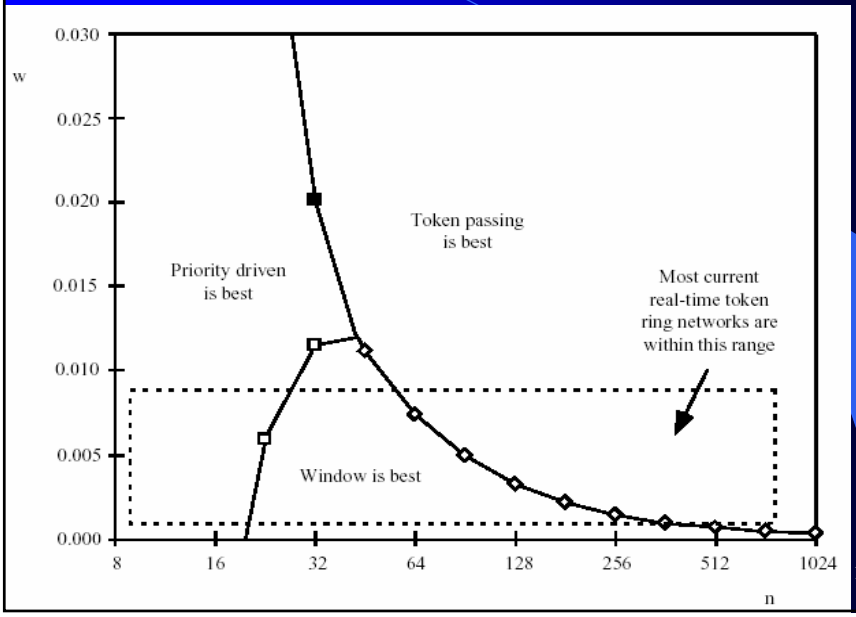
Georgia Tech

Scheduling Algorithms

- Rate monotonic for periodic jobs
  - Shorter period gets higher priority
  - Tight worst case achievable utilization
- Minimum laxity first for aperiodic jobs
  - Closer to failure gets higher priority
  - Optimal in minimization of failures
  - Related to EDF

Georgia Tech

Analytical Graph



Classification

- Scheduling algorithms fail at overload
- Prevent overload
  - Static schedules
  - Admission control
- Overload management
  - Congestion control
  - Adaptive (QoS-driven) resource management

Georgia Tech

Admission Control

- Asynchronous messages
  - Guaranteed dynamically
  - Conservative estimation: worst-case analysis of all messages in a node
  - Main problem: under-utilization
- How to sharpen the estimation and increase network utilization

Georgia Tech

Assumptions

- Network model
  - Diffserv architecture (with classes)
- Resource management components
  - **Configuration**: class & route determination
  - **Run-time admission control**: enforce admission policies when under overload
  - **Packet forwarding**: class-based static priority policy, FIFO within class

Georgia Tech

Utilization-Based

- Utilization-based admission control
  - Run-time decisions on flow establishment
  - Current utilization the only criterion
- Bandwidth availability along flow path
  - Safe levels of utilization
  - Determined during configuration

Georgia Tech

Conservative Estimation

- Configuration-time delay computation
  - *Traffic constraint* bounds usage per link
  - *Server delay bound*
  - Iterate the delay calculation on all servers
- Safe route selection
  - Satisfy deadlines of each class and route
  - Heuristics: min distance, min delay

Georgia Tech

Maximize Utilization

- Max utilization bounds depend on network diameter (theorem 4)
  - Given a utilization level, find safe routes
  - Converge on max utilization that has safe routes
- Their result (45%) is better than Shortest Path (33%)

Georgia Tech

Discussion

- What are the principles that guarantee RTES will work as designed?

Georgia Tech