Router Design:

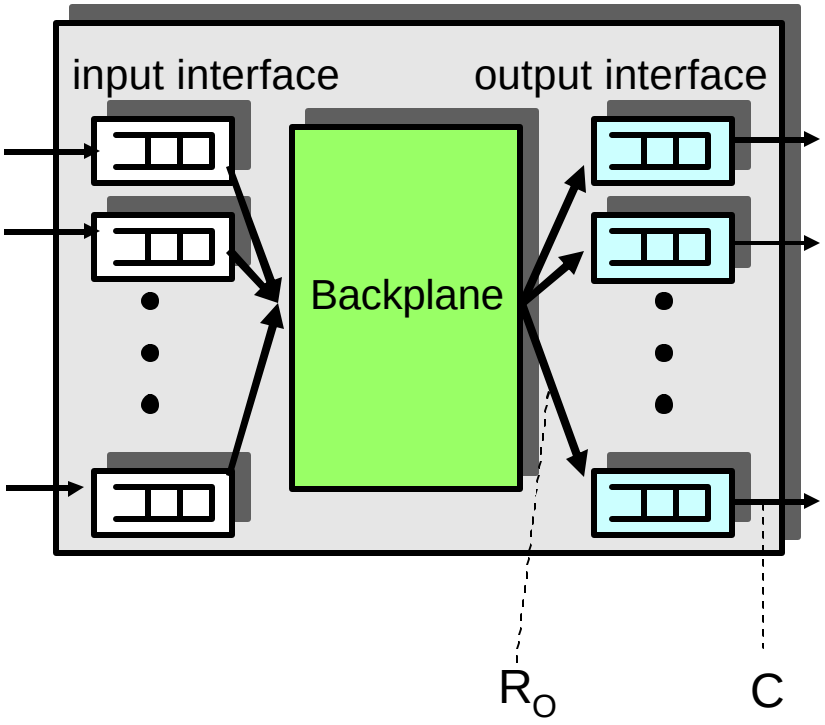
Table Lookups and Packet Scheduling

EECS 122: Lecture 13

Department of Electrical Engineering and Computer Sciences
University of California
Berkeley

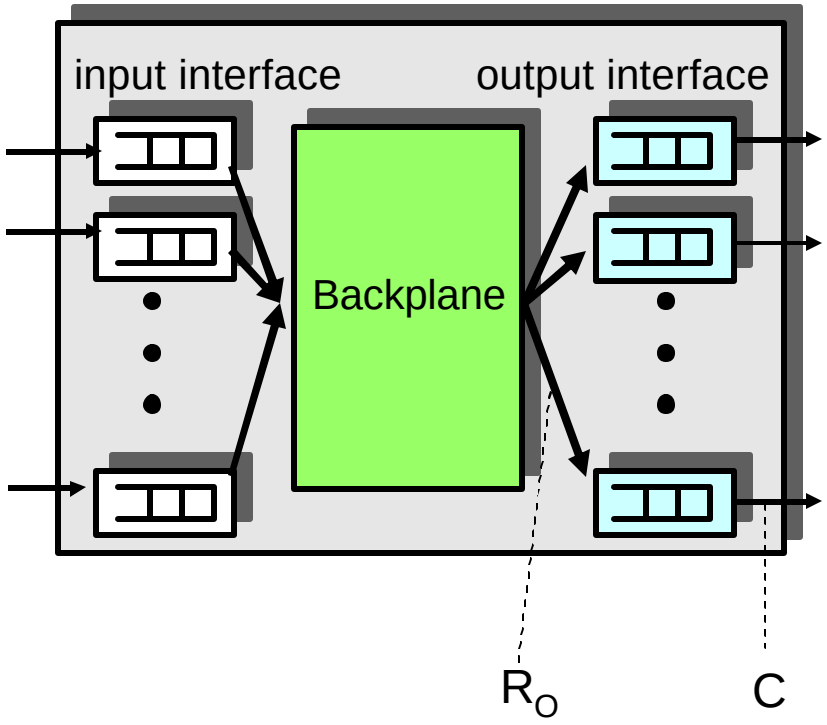
Review: Switch Architectures

- Input Queued
  - Faster switch
  - Two congestion points
    - HOL
- Output Queued
  - Slower switch
    - Backplane speedup is N
  - One point of congsestion



Today

- Fast routing table lookups
  - Exact and LPM
- How to resolve router congestion?
  - Assume output queued
  - Focus on scheduling
    - Already covered packet dropping
- Defer the end-to-end issues to next lecture

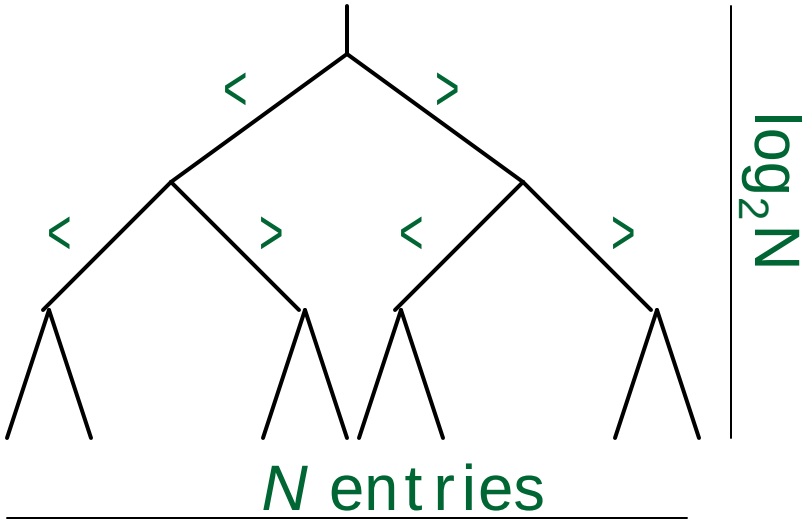


Route Lookup

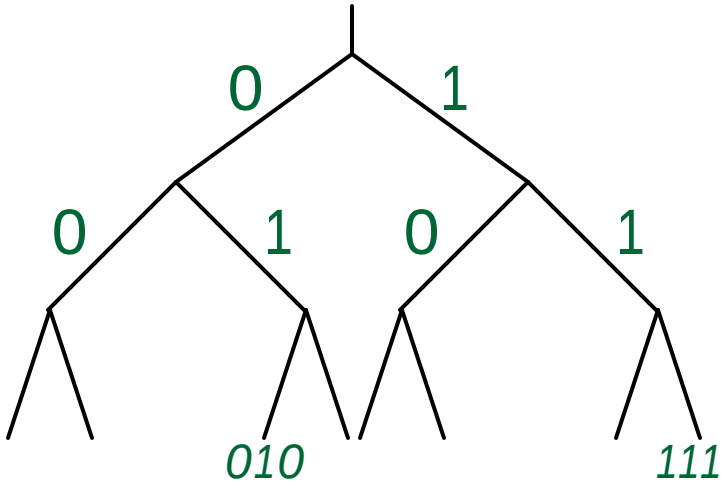
- Exact Match
  - MPLS/ATM: Direct lookup possible because of labels
  - Ethernet: Global address space is flat and huge
  - Hashing is an option but we won't discuss it…
- Longest Prefix Match
  - IP
- In Ethernet and IP routing it is useful to think of the search process as a traversal of a special kind of labeled tree called a Trie

Trees and Tries

Binary Search Tree



Binary Search Trie

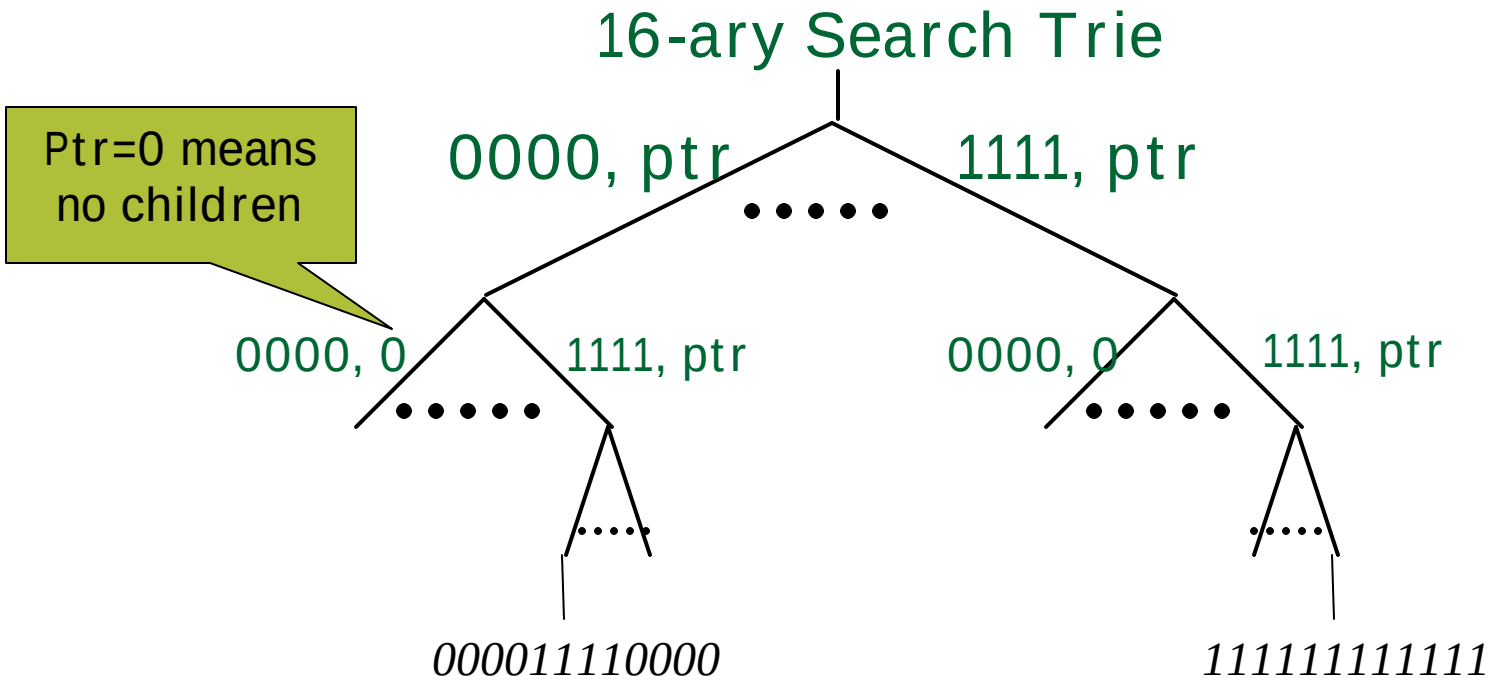


Nick Mckeown

Simple Tries and Exact Matching in Ethernet

- Each address in the routing table entry is of fixed length
  - E.g. using a simple binary search trie it takes 48 steps to resolve an MAC address lookup
  - When the table is much smaller than 2^48 (!), this seems like a lot of steps
  - Instead of matching one bit per level, why not m bits per level?

Multiway Tries



Nick Mckeown

Why not just have one level?

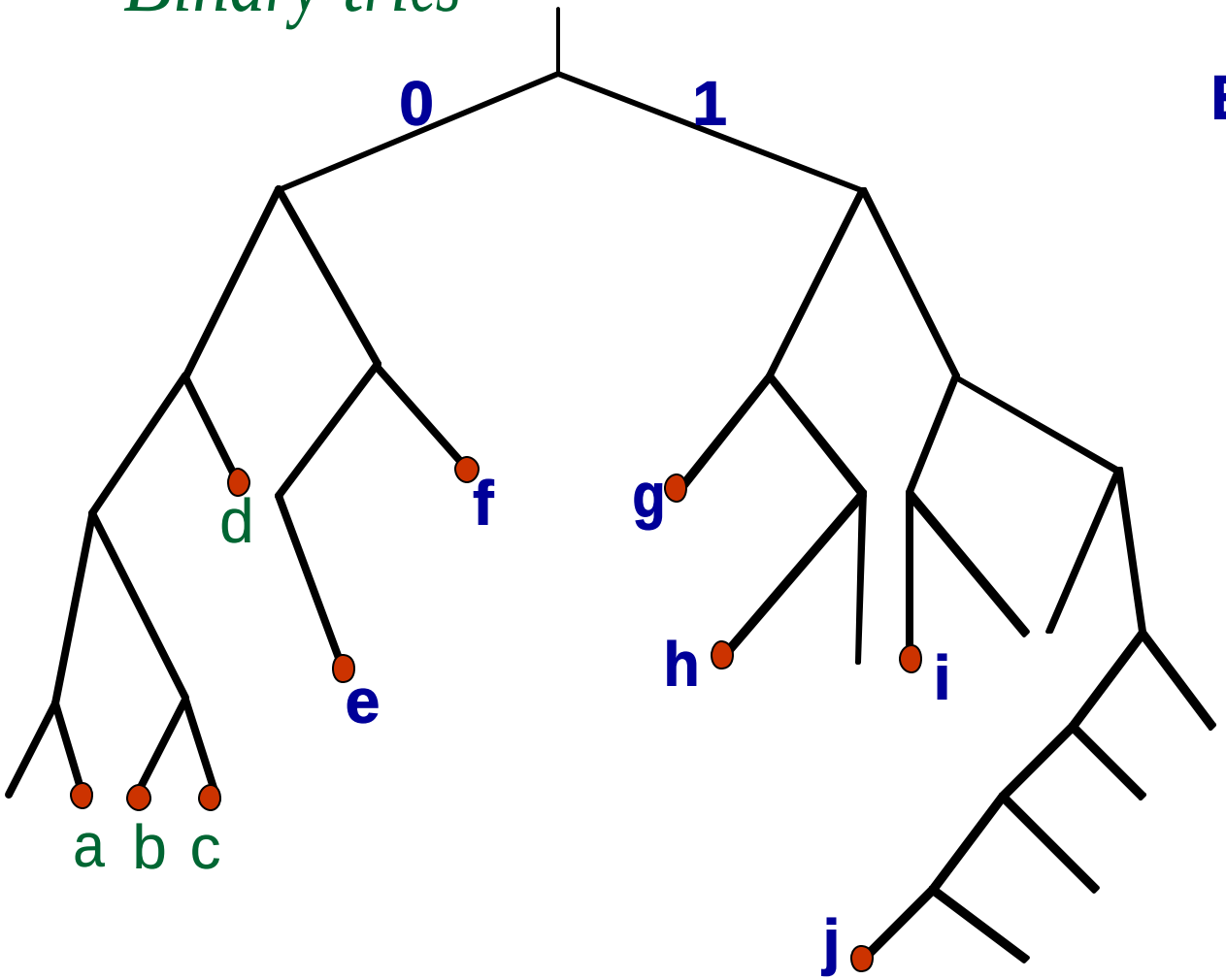
- Memory required goes up with higher values of m
  - Too many nodes with “0” pointers
  - Having more levels in the tree allows potentially large subtrees of the balanced tree to be to cut off and not stored
- The “right” number of levels is a function of memory constraints

Simple Tries and LPM

- The routing table entry is a variable length prefix
  - E.g. 01111111 00001111 0000100100 for 128.23.9.0/26
  - A balanced tree won't work
  - Variable number of steps required

LPM in IP Routers

Binary tries



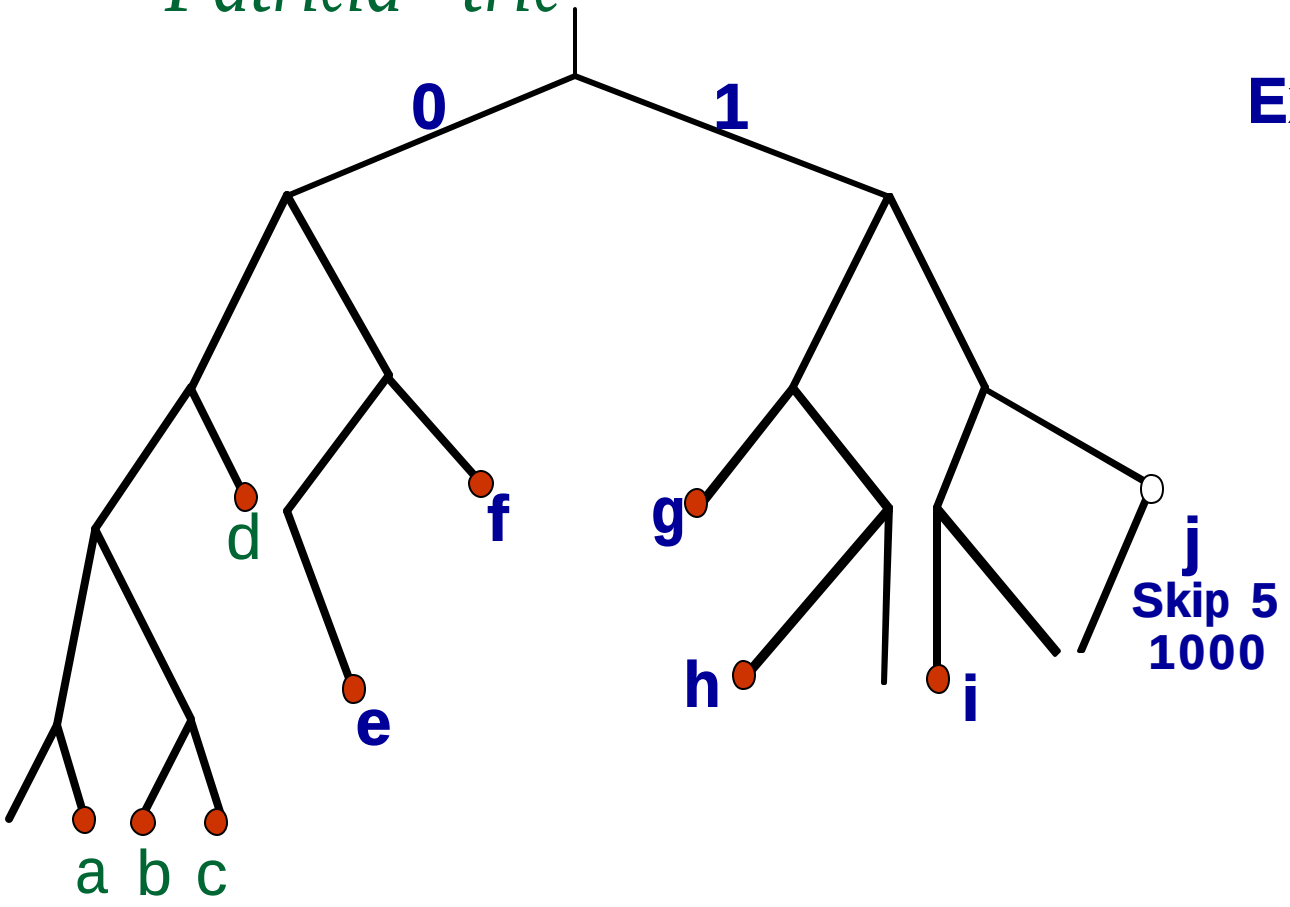
Example Prefixes

a) 00001
b) 00010
c) 00011
d) 001
e) 0101
f) 011
g) 100
h) 1010
i) 1100
j) 11110000

Nick Mckeown

LPM in IP Routers

"Patricia" trie



Example Prefixes

a) 00001
b) 00010
c) 00011
d) 001
e) 0101
f) 011
g) 100
h) 1010
i) 1100
j) 11110000

Nick Mckeown

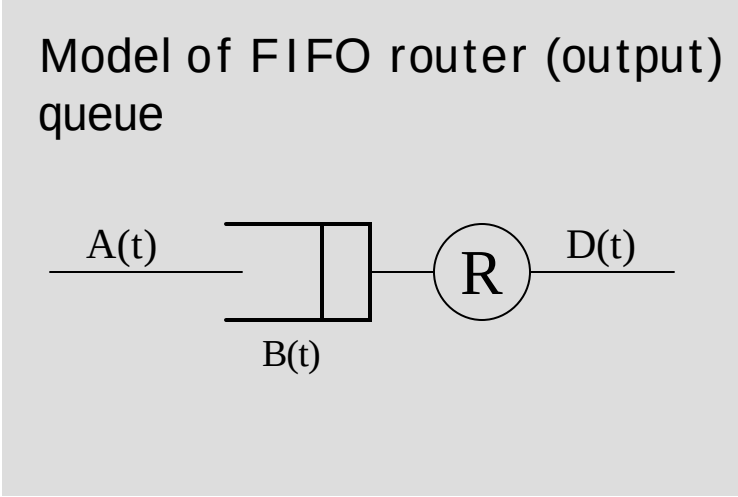
Associative Memory

- Content addressable memory is expensive, consumes power and is still slow, but makes the lookup problem easy
  - For Ethernet compares incoming data to all entries in parallel
  - For LPM do 32 exact matches in parallel
    - Match prefixes of length 1
    - Match prefixes of length 2
    - Etc.
  - Still not a practical, but this may change some day…

Scheduling and Classification

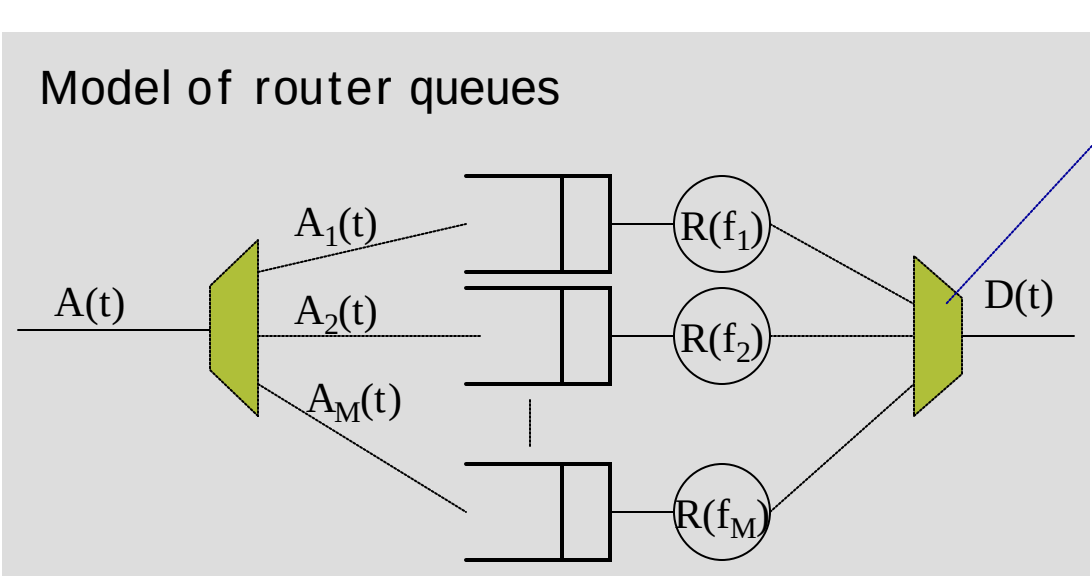
- In a store and forward network, contention for router ports can result in considerable packet delay
  - How do we manage this delay so that
    - The network does not get swamped with retransmissions
    - Certain kinds of applications are not “victimized” and bear an unfair share of the delay
    - Delay sensitive applications are protected
- Every packet is separated into one of M classes
- At each port, have a separate queue for each class
- Very similar (and complimentary to) issues addressed by RED and other packet dropping mechanisms

Router Without any classification



Classification and Scheduling

Router With M classes



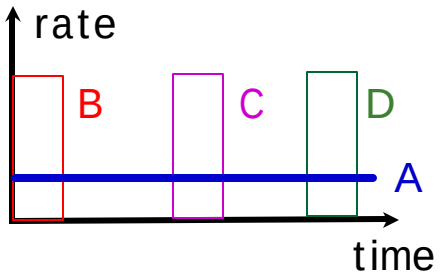
Nick Mckeown

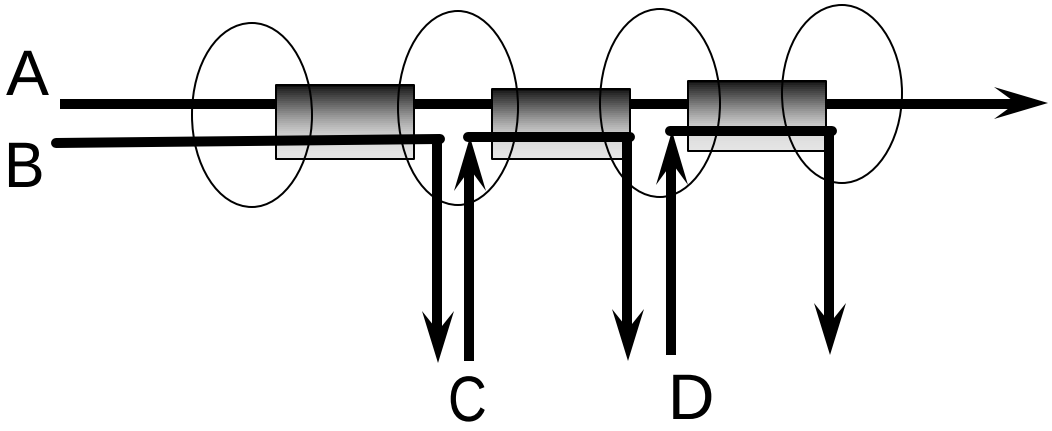
Classification

- Packet Class
  - A way of receiving differentiated treatment from the output port scheduler
  - Delay Sensitivity
  - Loss Sensitivity
  - Etc.
  - More fields ? harder the implementation
- How many queues?
  - More queues ? harder the implementation
  - One for each priority
  - One for each flow (example, VCI)
  - Etc.

Motivating Example: FCFS

- A is an audio session
  - Steady: (peak rate = avg. rate)
- B, C and D image files
  - Burst (high peak rate)

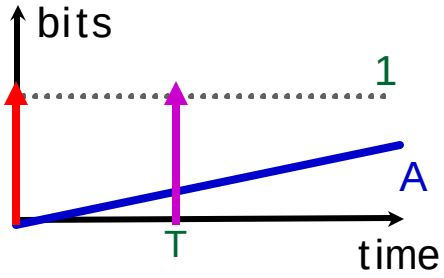




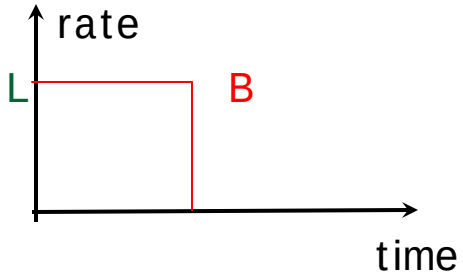
What happens to A?

Delays build up for the steady session

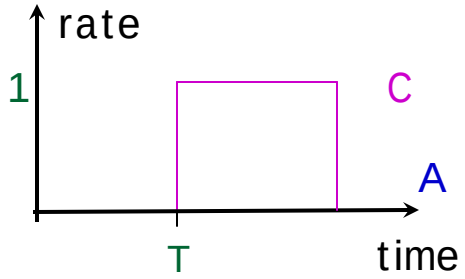
Arrivals



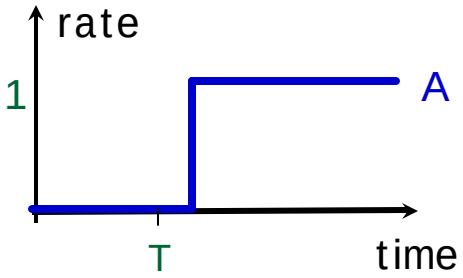
At Node 1

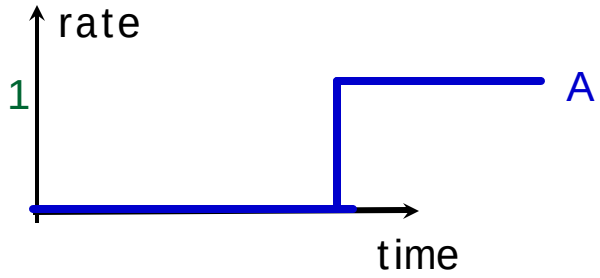


At Node 2



Both links FCFS at rate 1





Hard to control performance with FCFS

- Bursty traffic can unfairly hold up steady traffic
  - Makes steady traffic bursty in the process!
- Very difficult to analyze how a FCFS network will perform under reasonable traffic scenarios
- Only way to control performance is to make buffers very small
  - Difficult for high speed traffic

Fairness

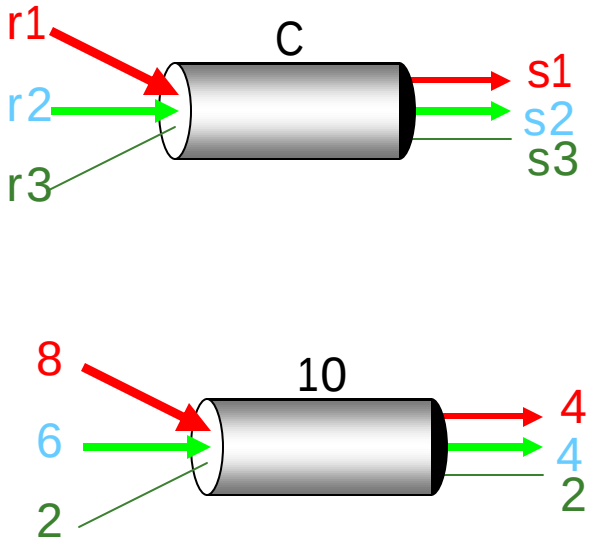
- Suppose there are N applications sharing the network
- Each application j can express its satisfaction with network performance in terms of $U_j(r)$, where $r = (r_1,\ldots,r_N)$ is the rate allocated to the applications.
  - Most common utility function is $U_j(r) = r$ for all j
- What is a fair allocation of the throughputs?
  - Fairness is vague so there are many definitions
    - Maximize the sum of utilities: May penalize some apps
    - Max-Min Fair

Max-Min Fairness

- Let an assignment be $(U_1(r),\ldots,U_N(r))$
- Given assignment, A, let MIN_A be the smallest component of the vector A
- Pick an assignment * such that for any other assignment, A
  - $W_* = MIN_A$
- There may be many max-min allocations
- Need to be able to classify packets by flow in order to implement
- Also needs a scheduling policy

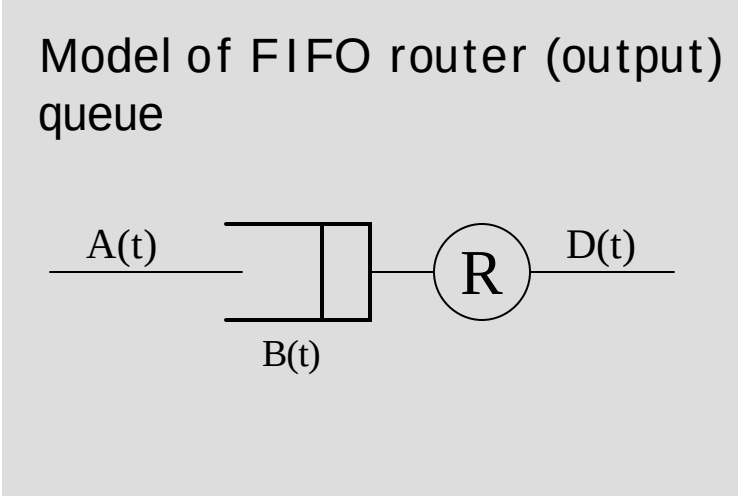
Simplest Case

- N flows, $U_j(r) = r_j$
- Max-min fair rate computation:
  1. Pick the unassigned flow, j, with the smallest rate
  2. $s_j = \min\{r_j, C/N\}$
  3. $C = C - s_j$
  4. N= N-1
  5. go to 1



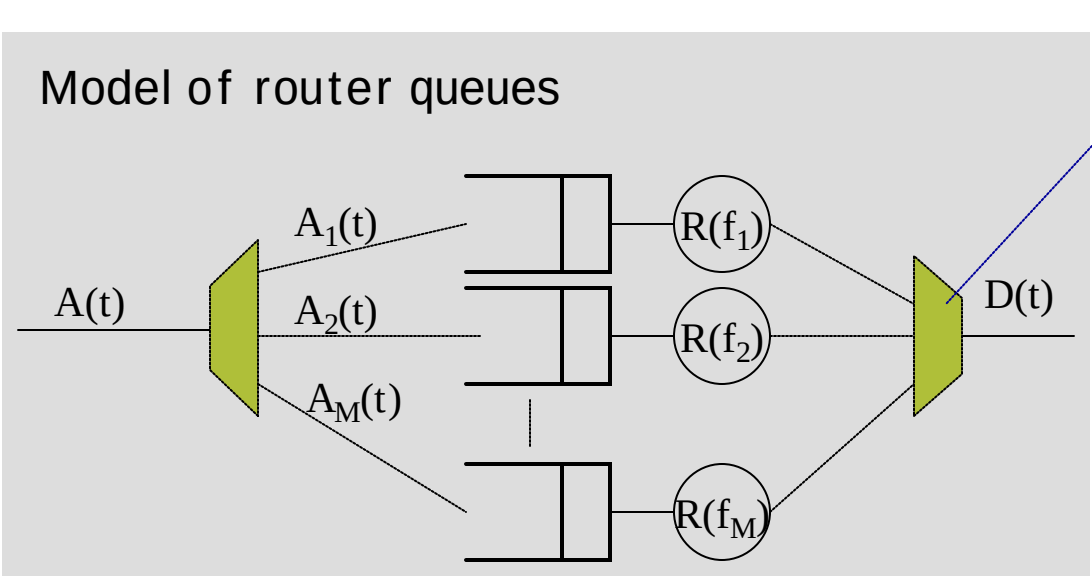
Why is this max-min fair?

Router Without any classification



Classification and Scheduling

Router With M classes



Nick Mckeown

Scheduling Goals

- Flexibility
  - Must be able to accommodate a wide range of performance objectives
  - Must be “fair”
- Predictable/Analyzable
  - Must have some way to determine if the performance objectives are met
- Implementable
  - Cost
  - Performance

Fluid Tracking Scheduling Policies

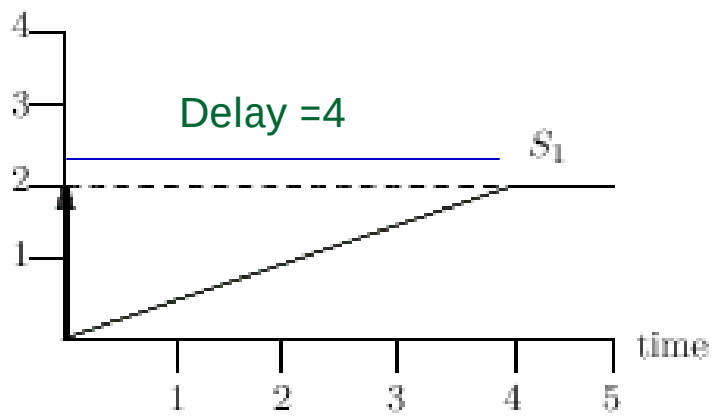
- The time taken to “serve” a packet of size L on a link of speed C is L/C.
- IDEALIZATION: Assume that L/C is infinitesimal
  - In any positive interval can serve traffic from any subset of the traffic classes
  - Traffic behaves as a fluid
- Design a fluid service discipline that is flexible and fair
- Approximate this ideal policy by a “Tracking” Service discipline with the true value of L/C

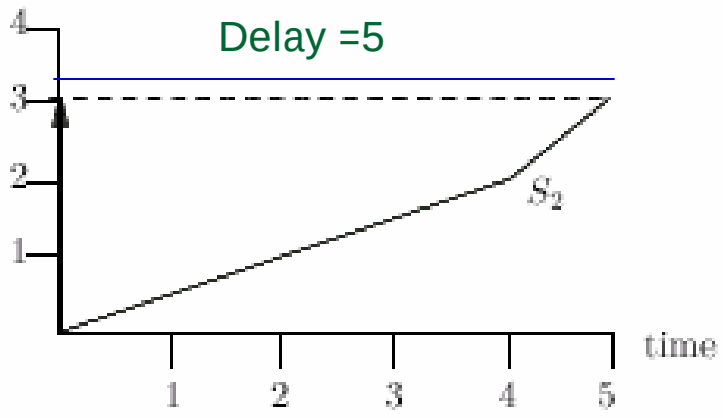
Generalized Processor Sharing

- Each class j has a weight w_j
- If there is traffic to be served from class j at time t we say that it is Backloged at time t.
  - B(t) is the set of backlogged classes at time t
  - W(t) is the total weight of the backlogged classes at time t
- If there is even one backlogged class, the server operates at rate C
  - This is called being work conserving
- Serve the classes in proportion to their weights
  - If class j is backlogged at time t, give it service rate
    - $S_j(t) = [w_j/W(t)] * C$

Generalized Processor Sharing

Class 1 . $w_1 = w_2$ Class 2





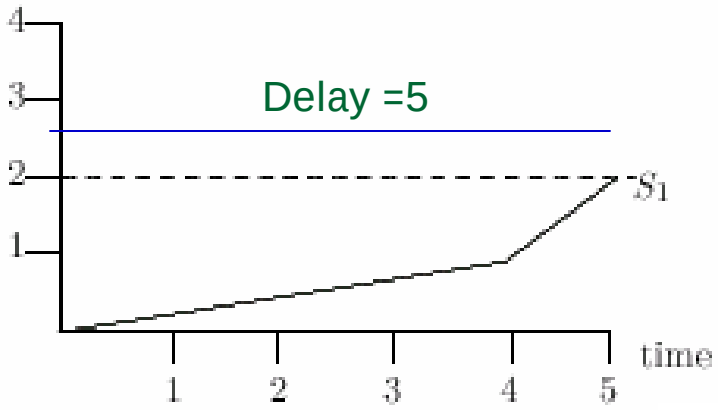
B(2)={1,2}
B(4)={2}

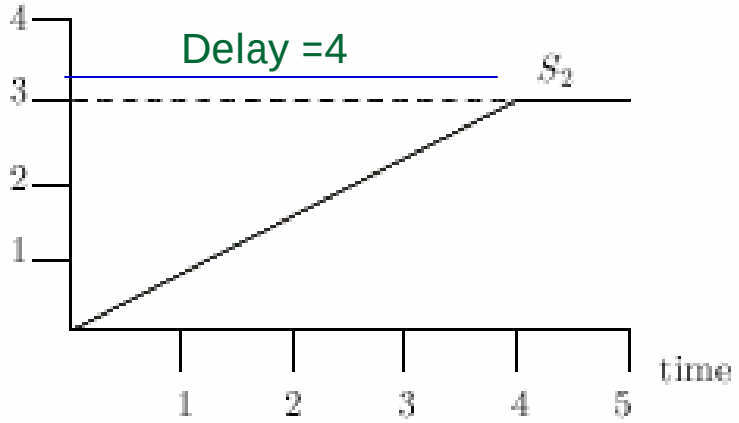
Adjust weights to change delay

Class 1

$3w_1 = w_2$

Class 2





B(2)={1,2}
B(4)={1}

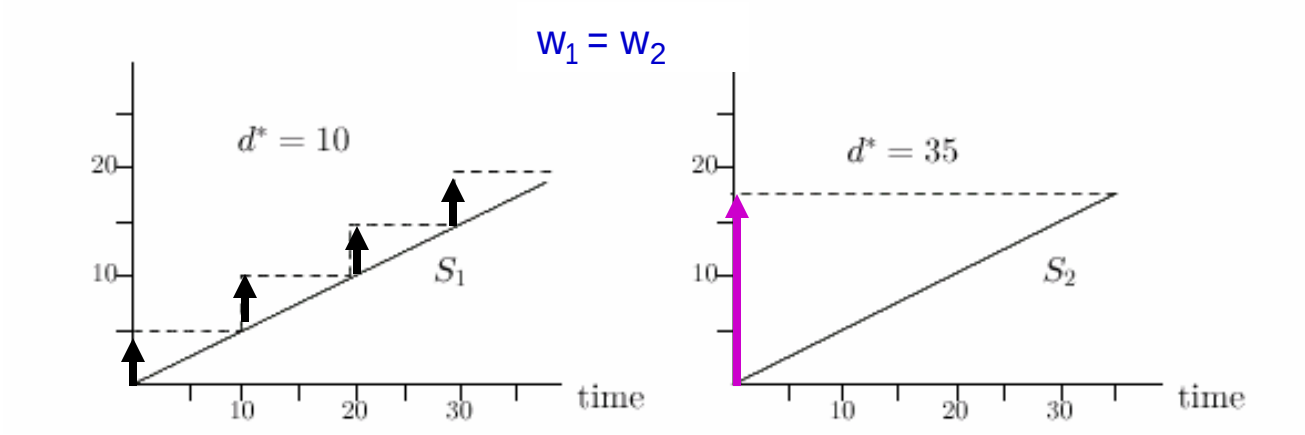
GPS Properties - I

- Let the sum of all the weights be F
  - F = W(t)
- When j is backlogged,
- $S_j(t) = [w_j/W(t)] * C$
- $= [w_j/F] * C = g_j$
- g_j is called the Backlog Clearing Rate
- If an arriving class j bit sees a queue of q class j bits it must leave in q/g_i sec.
  - This is an example of a delay bound
- Suppose audio class comes in at 64Kb/s and has BCR = 80Kb/s
  - Set $w_j = (80000 * F)/C$
  - Performance should be good regardless of how the other classes arrive
    - Can't do this under FCFS

GPS Properties - II

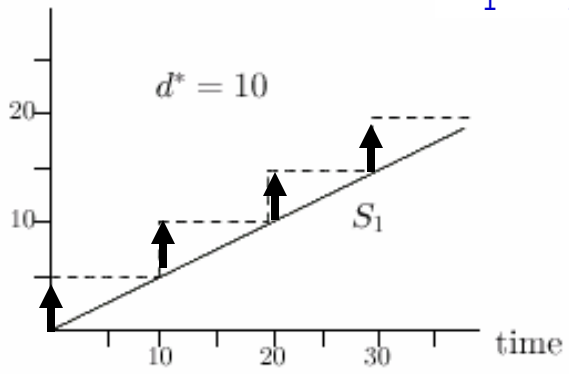
- If a class has an average rate of r and we give it a weight so that $g_j < r_j$ then does the class j queue blow up?
  - Not if the aggregate incoming rate (rate summed over all classes) is less than C
    - If it isn't some queue has to blow up under any service discipline
    - Finding a delay bound is trickier
- Giving steady classes a BCR much higher than r_j keeps them steady and doesn't hurt other classes

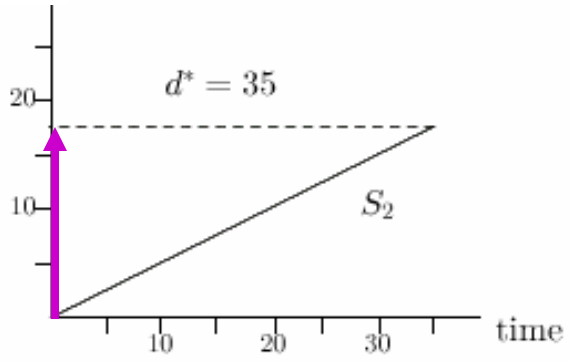
Treat Steady Sessions Well!



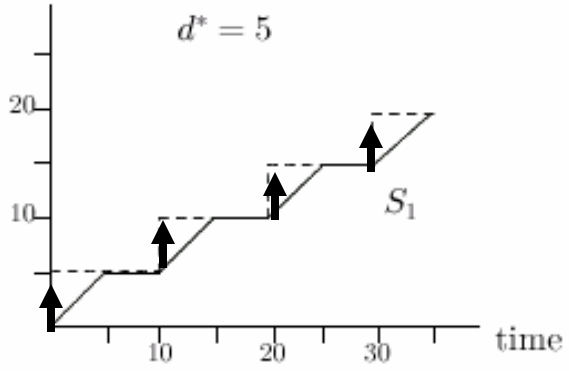
Treat Steady Sessions Well!

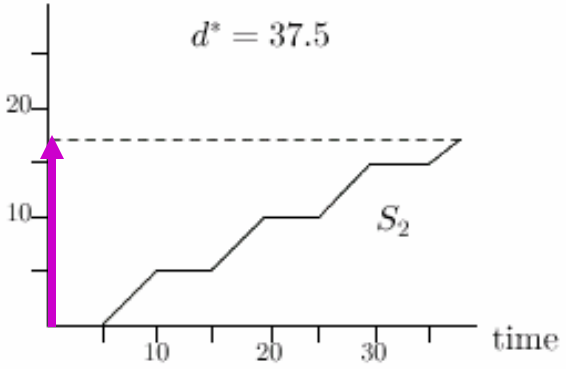
$w_1 = w_2$





$w_1 >>>> w_2$

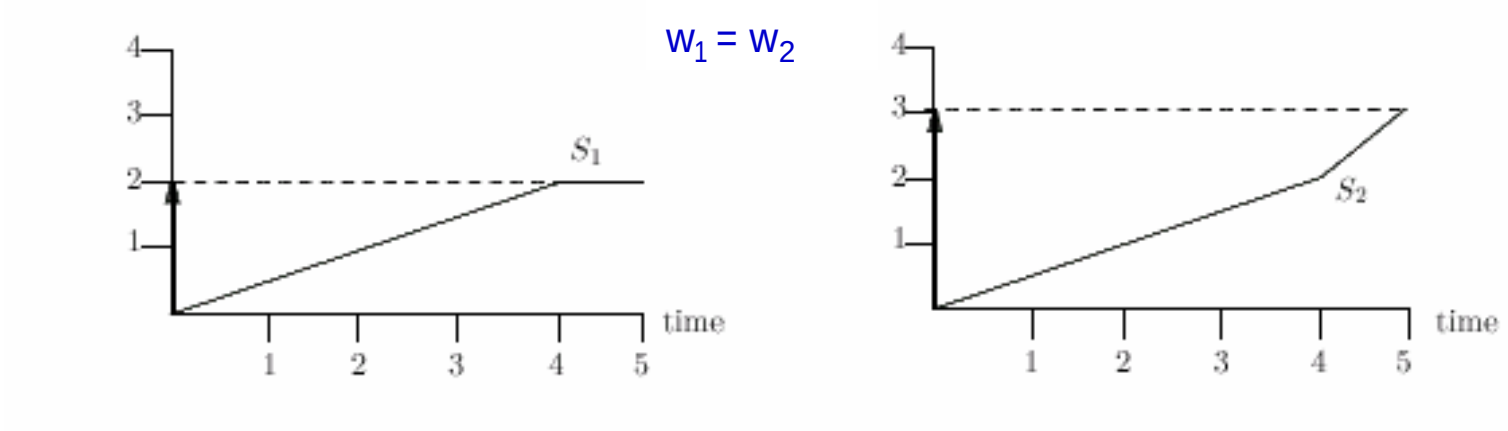




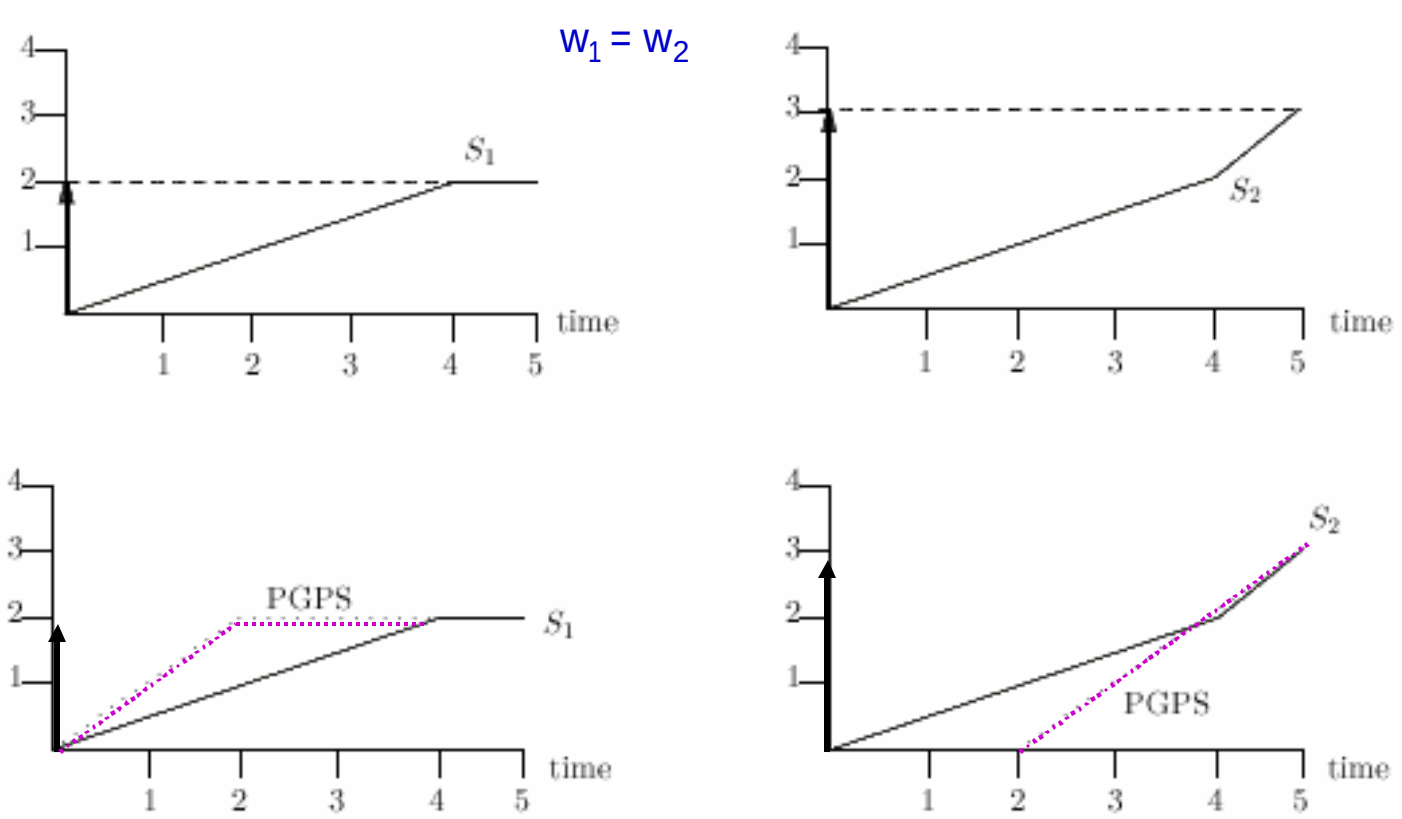
Weighted Fair Queueing

- Work Conserving
- Tracks GPS
  - Simulates the arrivals in real-time in a GPS system.
  - Serves entire packets (only one class at a time)
  - Let F_p be the time that the last bit of packet p departs the simulated GPS system
  - WFQ attempts to serve packets in order of F_p
  - Also called “Packetized Generalized Processor Sharing” (PGPS)

WFQ/PGPS Example



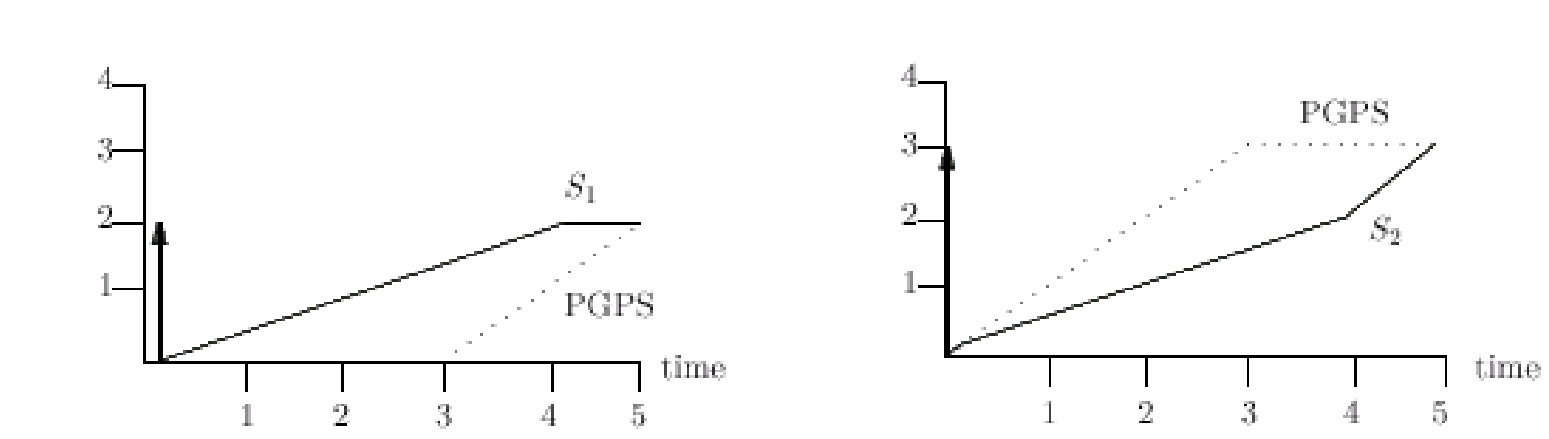
WFQ/PGPS Example: Fair Queueing



No packet finishes later under PGPS

WFQ Example II: Fair Queueing

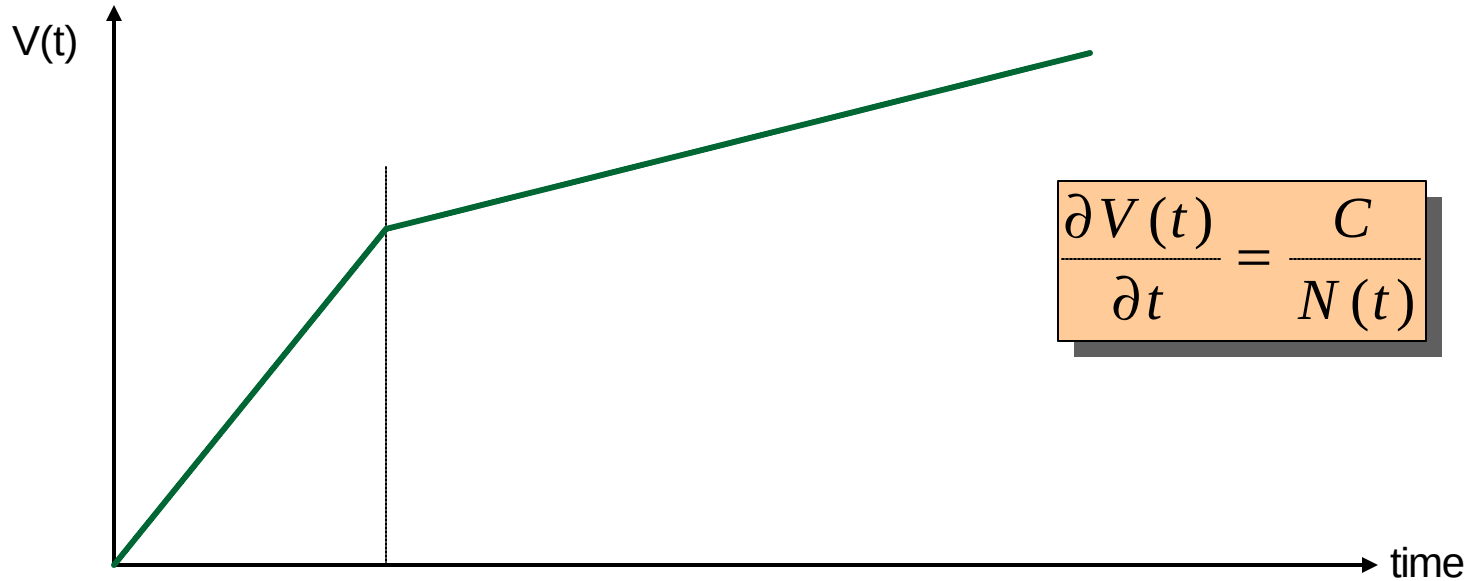
$w_1 = w_2$

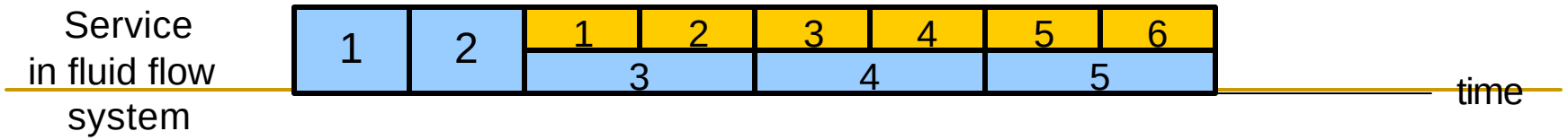


- Packets out of order
  - Packet from Class 1 finishes late under WFQ
  - Result: Maximum lateness is L_{max}/C where L_{max} is the maximum packet size allowed. WFQ never falls further behind.

Virtual Time Implementation (FQ)

- Measure service, instead of time
- V(t) slope – rate at which every active flow receives service
  - *C* – link capacity
  - *N*(t) – number of active flows in fluid flow system at time t





Fair Queueing Implementation

- Define
  - F_i^k finishing time of packet k of flow i (in system virtual time reference system)
  - a_i^k arrival time of packet k of flow i
  - L_i^k length of packet k of flow i
- The finishing time of packet $k+1$ of flow i is

$$F_i^{k+1} = \max(V(a_i^{k+1}), F_i^k) + L_i^{k+1}$$

WFQ and Performance

- Since GPS is analyzable, WFQ can be analyzed as well using the relationships between the two
  - This allows for predictable performance in the network
- WFQ implemented on most routers
- For core fast routers which are input queued more complicated tracking policies for GPS must be devised
- Meaning of class:
  - Session
  - Type of traffic
  - CEO's traffic
  - Etc.

Scheduling disciplines abound!

- Many improve WFQ
  - Eg. WF^2Q, Hierarchical FQ etc
- Some are much easier to implement
  - Deficit Round Robin
- Non-Work Conserving alternatives
- Earliest Deadline First
  - Each packet has a deadline – not a class
  - Most “flexible” service discipline

Summary

- Scheduling is an important mechanism to implement a policy on how to share the bandwidth of an output port
  - Mainly used to manage delay
- Scheduling is only effective if packet buffers are large enough to matter
  - Overprovisioning argument (popular but controversial)
  - Virtual queues in TCP may make this small
- Scheduling should be used in conjunction with packet dropping

Next Lecture

- The effect of scheduling on end-to-end performance
- Signaling
- QoS/CoS