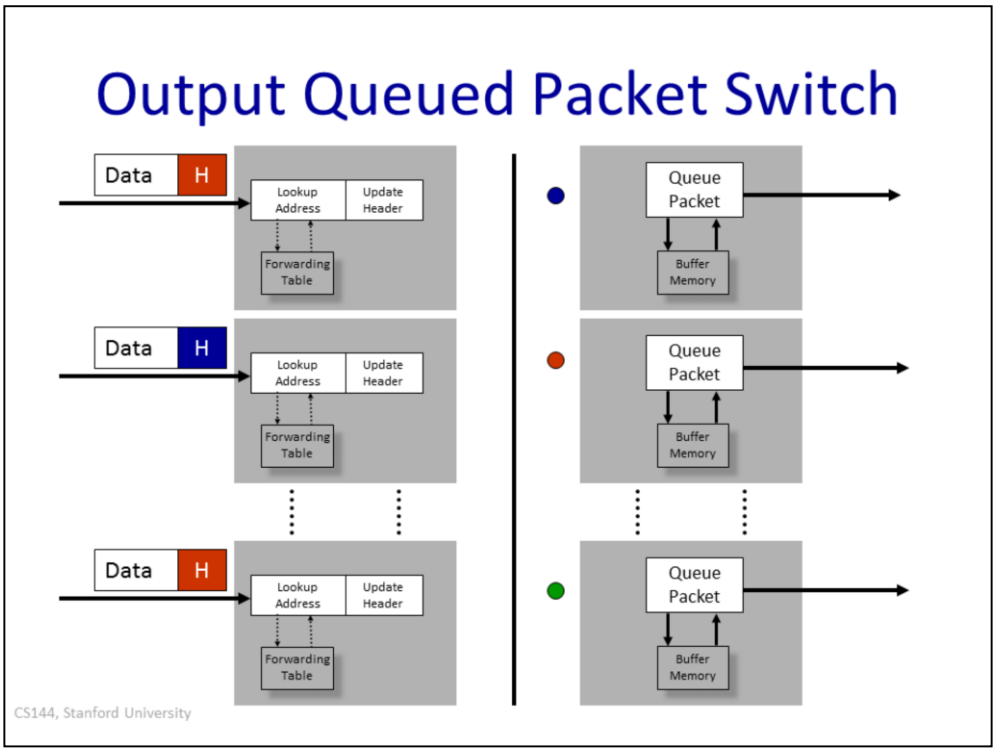
CS144
An Introduction to Computer Networks

Packet Switching

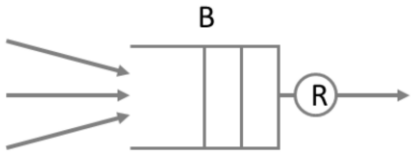
Strict priorities and guaranteed flow rates

Nick McKeown
Professor of Electrical Engineering
and Computer Science, Stanford University



Run animation. Say that we're going to be homing in on the output queue. DRAW circle around output queue.

FIFO is a free for all



There is a Free For All in the queue. If there are many flows passing through the queue, they will depart in FIFO order. Whoever sends the most will get the highest rate. If there is a really big hog of a flow, a little flow could get squeezed out completely. Encourages bad behavior – the best thing for a flow to do is try and crowd every other flow out by sending as fast as it can. Not very friendly behavior…Not good incentives.

Now imagine that some traffic was very urgent – for example control traffic, or some important video traffic. The FIFO doesn't have any way to distinguish importance. It just says: If you got here first and there was room in the queue, you are most important packet.

We can't say anything meaningful about the rate of each flow sharing the queue.

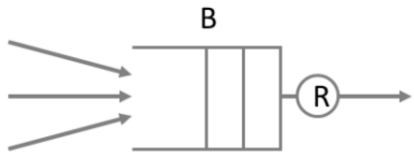
Notice that if a packet does make it into the queue, the maximum time it has to wait is B/R. We'll use this observation later.

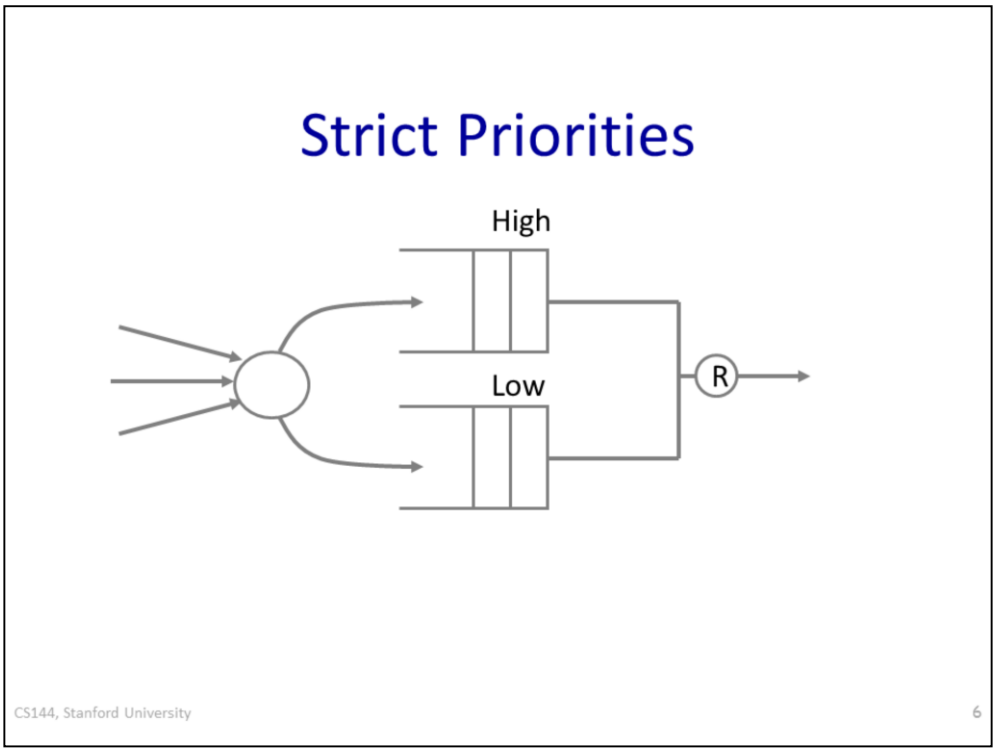
Outline

1. Strict priorities
2. Rate guarantees

In this video, we're going to look at how we can give higher priority to some flows than others; and how we can give each flow a different service rate.

Single FIFO





Imagine that we replace the single FIFO queue at an output with two priority queues; A high priority queue and a low priority queue.

When a packet arrives, we decide which queue to put it in based on how important it is.

Then – because it is a strict priority – the rule is that we always serve the high priority queue first. We ONLY serve the low priority queue if the high priority queue is EMPTY.

The consequence is that the High priority traffic doesn't see the low priority traffic – it is unaffected by it. It's as if the high priority traffic has its own private network.

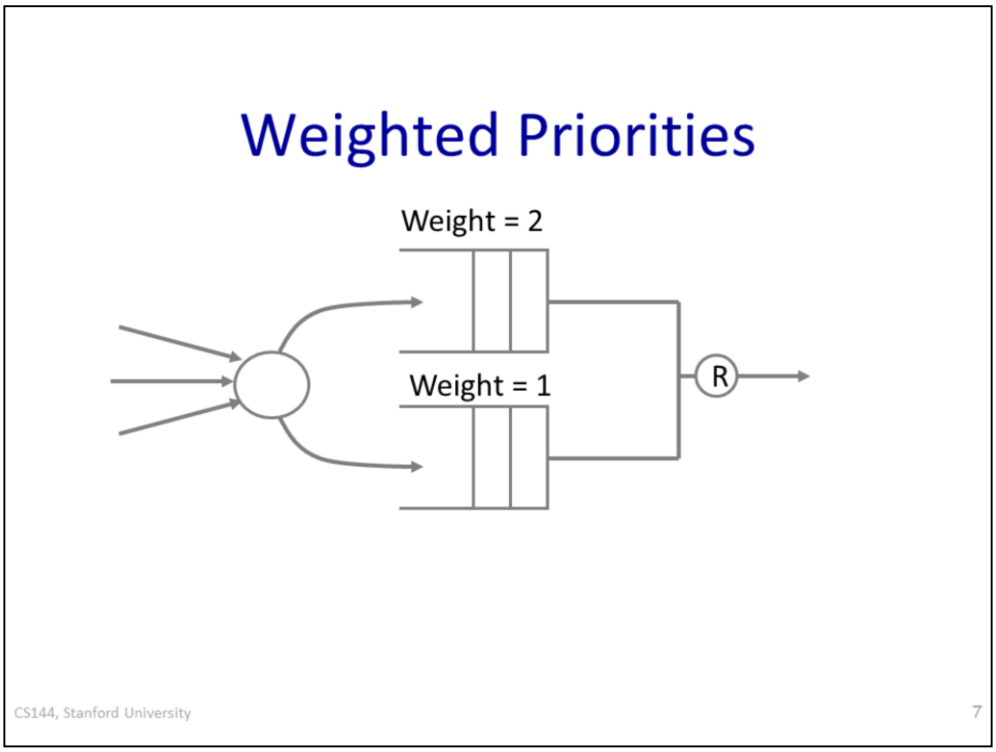
Strict priorities are quite commonly used, and most switches and routers support them today.

A common way to determine which queue a packet is to be placed into is to mark the header with its priority. For example, the IP header has a field called ToS or TYPE of SERVICE, where we can mark the priority of the packet.

It could be used, say, to makr VIDEO traffic > Email. Or Control > DATA. Or

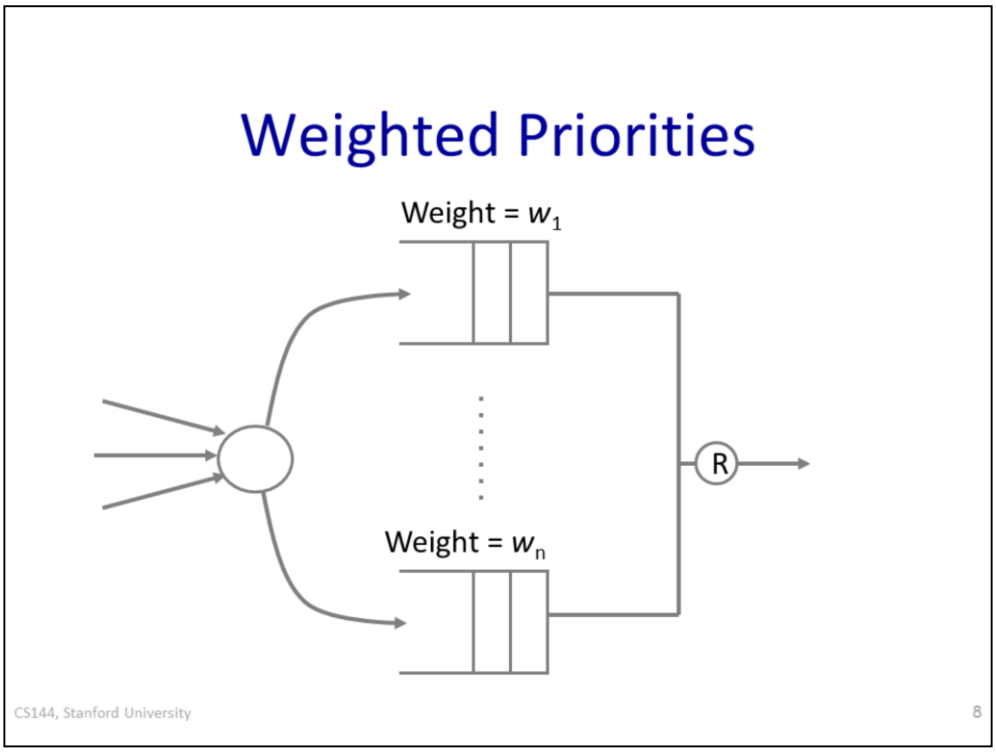
Gold user > Silver User.

OK for some things. But it starves out the LOW priority traffic, so you only use it when there is reasonably small amount of high priority traffic. We don’t want to use this if the High Priority traffic can completely hog the link.



So instead, we might want to do something like this, where the traffic in one queue is weighted more than the traffic in the other queue; for example, the traffic in the upper queue can be served at TWICE the rate of the traffic in the lower queue.

We can generalize this to the situation where there are “n” queues, each with their own rate……

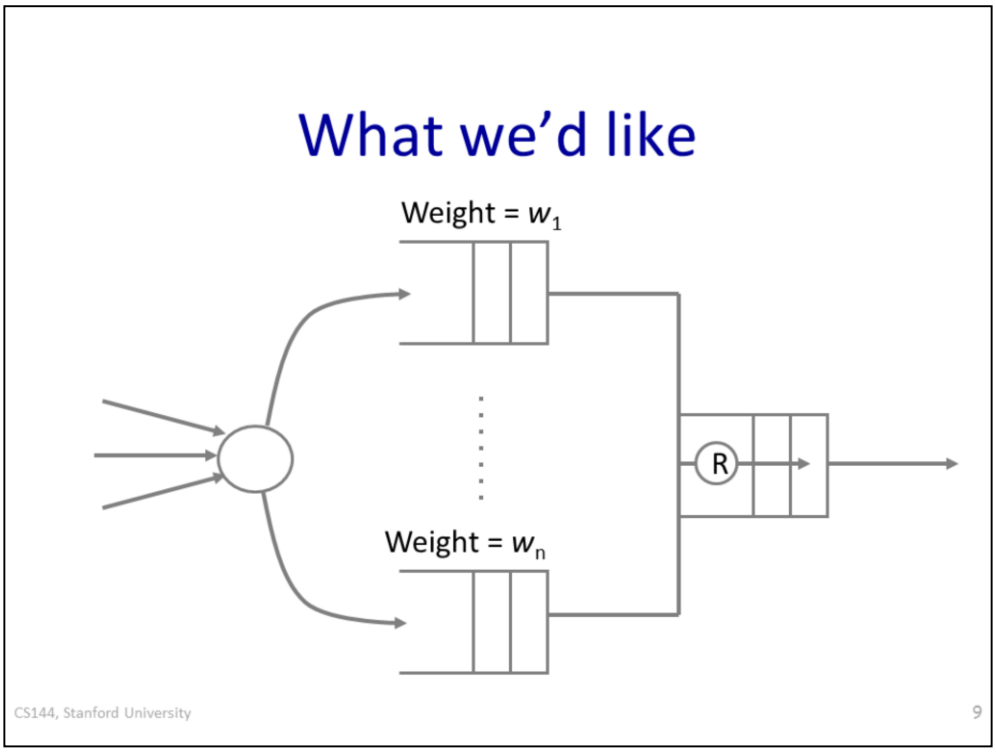


How can we accomplish this in practice???

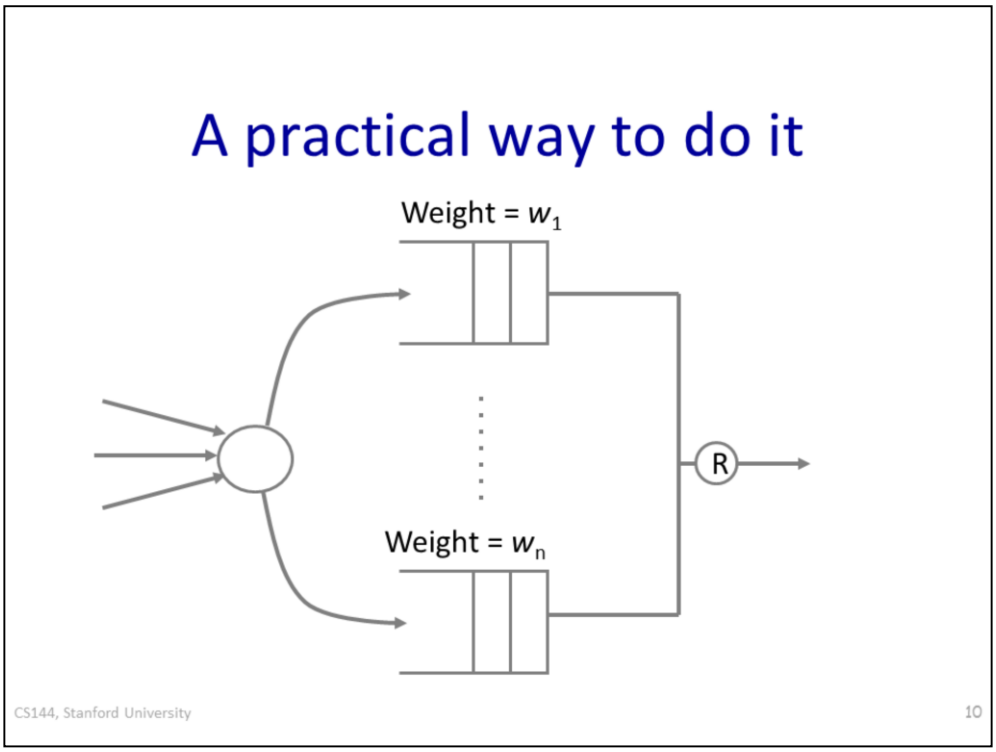
If all the packets were the same length, then this would be easy. We would serve: w1/ Sum(wi) packets from the first queue, then w2/(Sum()) form the 2nd and so on, then repeat.

DRAW bits on the wire according to weights.

This is all very good, but real packets are VARIABLE length. They might be as short as 64bytes, or as long as several thousan bytes – i.e. a variation of 2-3 orders of magnitude. Clearly we MUST take into account the packet lengths if we want to prevent long packets crowding out short packets.



WE could go in ROUNDS and serve each queue with up to w_i bits in each round. That way, they get the right weighted service. Then, we can use a queue to construct full packets from all the fragments and send them out.



"PRETEND" we serve each packet bit by bit.

Figure out what time it would finish if we *did* serve bit by bit.

Then serve all the packets in the order they *would* have finished under bit by bit.

Nice simple way to do this – very clever:

* Define a ROUND to be the time to visit and serve every queue.

* In each round, serve queue I with w_i bits.

* Do the following calculation all in terms of the ROUND NUMBER:

When packet k arrives: F_k = Max(F_(k-1), now) + L_k/w_k

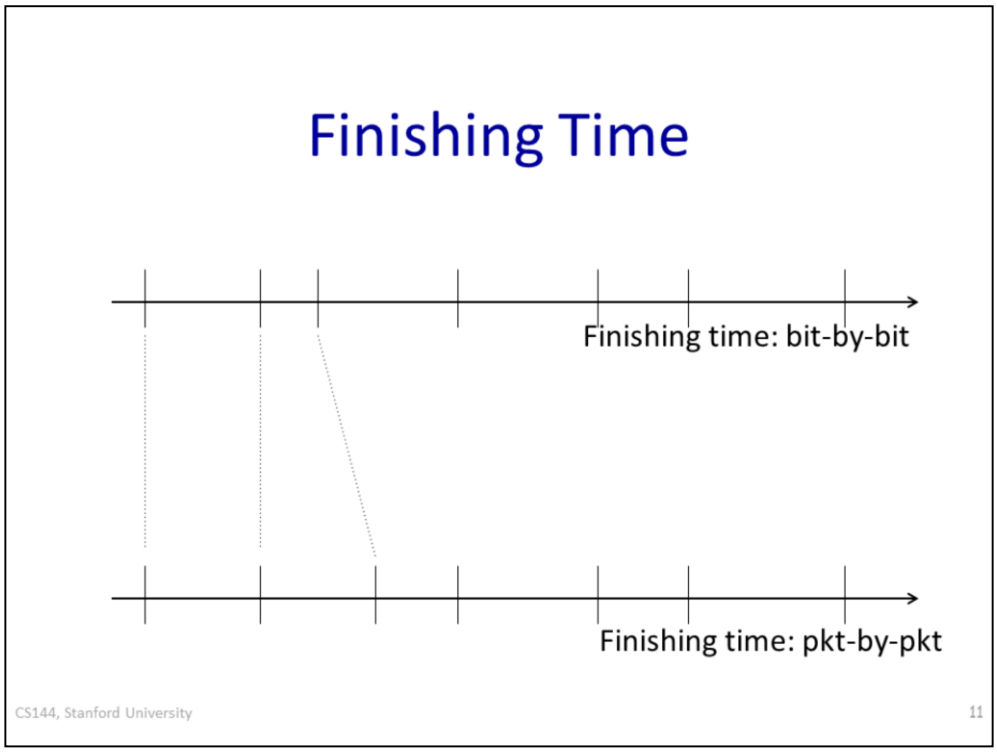
Now serve the packets in order of F_k, the FINISHING ROUND.

Three nice properties:

1. Finishing times can be determined at the packet arrival time. T

2. The packets are served in order of finishing time.

This approach is called Weighted Fair Queueing (WFQ) or Packetized Generalized Processor Sharing (PGPS).



Sketch time lines of FINISHING ROUND -> FINISHING TIME.

Sketch on it that the difference, DELTA, between the finishing time when calculated using ROUNDS and the actual finishing time $< L_{max}/R$ for EVERY packet in the system.

This means that over the long term, the DELTA is amortized, and the rate of each flow is simply: $w_i/(\text{Sum}(w_i)) * R$.

Summary

FIFO queues are a free for all: No priorities and no guaranteed rates.

Strict priorities: High priority traffic "sees" a network with no low priority traffic. Useful if we have limited amounts of high priority traffic.

Weighted Fair Queueing (WFQ) lets us give each flow a guaranteed service rate, by scheduling them in order of their bit-by-bit finishing times.