01 - Router Design  
  
In this lesson, we will cover the design of big, fast, modern routers. Here's a picture of two modern router chassis. On the left, we have a picture of the Cisco CRS-1, and on the right, we have a picture of the Juniper M320. The M320 has, for some time, been used at the border of the Georgia Tech network between Georgia Tech and the rest of the Internet. Here's a picture of a couple of line cards that go into these chassis. These kind of look like network interface cards, except the ports are special, instead of terminating ethernet, these ports terminate high capacity fiber links. As you can see, these cards are actually a whole lot bigger than your typical network interface card as well. And, as a result, these chassis are often anywhere from three to six feet tall, and can fill up an entire rack. There's a significant need for big, fast routers. Links are getting faster. Traffic demands are also increasing, particularly with the rise of demanding applications, such as streaming video. Networks are getting bigger too, in terms of the number of hosts, routers, and users. So there's a perennial need to design big, fast routers, particularly in Internet backbone networks. The rest of this lesson will focus on how a router works, in particular, how it goes from the process of taking a packet as input and sending it on to where it needs to go. The Internet's routing protocols, of course, are responsible for populating the forwarding tables on a router. But once those tables are populated, the router still has the hard job of taking a packet as input and ultimately getting it to the appropriate output port, so that the traffic can proceed and route to the destination.

02 - Basic Router Architecture  
  
Let's take a look at a generic router architecture. As a summary of basic router function, a router receives a packet. It then looks at the packet header, to determine the packet's destination. It looks in the forwarding table, to determine the appropriate, output interface for the packet. It modifies the packet header, such as decrementing the time to live field and updating the IP header check sum appropriately. And finally, it sends the packet to the appropriate output interface. The basic I/O component of a router architecture is the line card, which is the interface by which a router sends and receives data. When a packet arrives, the line card looks at the header to determine the destination, and then it looks in the forwarding table. To determine the output interface. It then, updates the packet header, and finally, sends the packet to the appropriate output interface. Now this drawing shows just a single line card. But in fact, when the packet is sent to the output interface, it must traverse the router's interconnection fabric, to be sent to the appropriate output port. So in fact, we can zoom out from that depiction of a single line card, and what we have, is a bunch of line cards, that are all connected via an interconnection fabric. Each of these line cards has a lookup table, the capability to modify headers, and a queue, or buffer, for packets, as they enter and leave the line card. In other lessons, we talk about several important questions. Such as how big queues should be and how lookup works. In the rest of this lesson, I'll discuss important decisions in router design. Such as, the placement of lookup tables on each line card and the design of the interconnection fabric.

03 - Each Line Card Has Own Forwarding Table Copy  
  
One important decision in the design of the modern routers was to place a copy of the forwarding table on each line card in the router. Well, this introduces some complications in making copies of the forwarding table. Doing so, prevents a central table on the router from becoming a bottleneck at high speeds. Consider an alternative where the router only has one copy of forwarding table. . In that case all of the line cards would need to be performing look ups on a central table. Which involves communication across the back plane as well as many more look ups against a central table. So while distributing the forwarding table across line cards prevents a central table from becoming a bottleneck. Early router architectures did not place the look up table on each line card, and as a result, when packets arrived at an individual line card, they would induce a look up in a shared buffer memory which could be accessed over a shared bus. But this shared bus, of course, introduces a bottleneck as well as contention between the different line cards that may be all performing lookups to the same shared memory. The solution, of course, was thus to remove the shared memory and instead place copies of the forwarding table on each line card. In summary, An important innovation in the design of these router was to eliminate the shared bus and place the look up table on individual line bus.

04 - Decision Crossbar Switching  
  
The second important decision is the design of the interconnect, or how the line cards should be connected to one another. Now one possibility is to use a shared bus. But the disadvantage of a bus for the interconnect is that it can only be used by one input-output combination. In any single time slot. What we'd like to do is enable input output pairs that don't compete to send traffic from input to output during the same time slot. For example, one should be able to send to four, two to six and three to five, all in the same time slot. The solution to this problem is to create what's called. A crossbar switch, or sometimes is also called a switched backplane.

05 - Crossbar Switching  
  
In crossbar switching every input port has a connection to every output port, and during each time slot, each input is connected to zero or one outputs. The crossbar is often depicted as follows. So if one wants to send to four, we could connect the input to the output in that time slot, and now this row and this column is occupied. But we could connect two to six and three to five in the same time slot without introducing contention. So the advantage of this design is that it can exploit parallelism by allowing multiple packets to be forwarded across the interconnect in parallel. But of course we also need proper scheduling algorithms To ensure fair use of the crossbar switch. Let's take a quick look at what this algorithm needs to achieve.

06 - Switching Algorithm Maximal Matching  
  
We'd like the cross bar switching algorithm to achieve what's called a maximal matching. Conceptually we have a router with n inputs and n outputs, but of course the inputs are also outputs. It's just easier to think about the inputs and the outputs being separate when we talk about the switching problem. Now in each time slot, we would like to achieve a one-to-one mapping between inputs and outputs, which is a matching. And our goal is that the matching should be maximal. So in a particular time slot, we might have a certain set of traffic demands, or traffic at certain input ports, that is destined for certain output ports. And our goal is, given these demands, to produce a matching that is maximal and fair. Now, given demands for a particular time slot and the resulting matching, notice that certain demands were not satisfied. These packets that arrived at inputs must wait until the next time slot to be forwarded to the appropriate output port, because they couldn't be matched in the same time slot as those shown here. Remember that there must be exactly a one-to-one matching between any inputs and outputs in a particular time slot. Most router crossbars have a notion of speedup whereby multiple matchings can be performed in the same time slot. So for example, if the line cards are running at, say, ten gigabits per second, then running the interconnect twice as fast would allow matchings to occur twice as fast as packets would arrive on the inputs or be forwarded from the outputs. It is thus common practice to run the interconnect at a higher speed than the input and output ports. Just speeding up the interconnect does not solve all problems. Note, for example, that in this set of demands we have packets arriving at this input port destined for this output port, but if there's only a single queue at this input, the packets that are destined for the output port, circled in orange, might actually be blocked behind a set of packets that are destined for other output ports. So even if we could induce a speed up at the interconnect, certain packets may be blocked in the queue by packets ahead of them destined for other output ports.

07 - Head of Line Blocking  
  
For example, if we have packets arriving in this queue destined for the orange queue, at the front of the queue, then even with the speed up, there may be packets that are sufficiently far behind in the queue that they're waiting behind the orange packets. What we'd like to be able to do is perform matchings to allow these packets to be sent to the output ports, and not have to wait for the entire queue to be drained of packets destined for the orange output port. A solution is to create virtual output queues, where instead of having a single queue at the input, we have one queue per output port. This prevents packets at the front of the queue that are destined for a particular output port from blocking packets that could otherwise be matched to other output queues in earlier timeslots.

08 - Scheduling and Fairness  
  
Let's now talk about scheduling and fairness. And then we talked about, in a crossbar switch, the process of matching input ports to output ports. The decision about which ports should be matched in any particular time slot is a process called scheduling. There are two important goals in scheduling. One is efficiency which is to say that if there is traffic at inputs distant for output ports, the crossbar switch should schedule inputs and outputs. So that traffic isn't sitting idle at the input ports if some traffic could be sent to the available output ports. Another consideration in scheduling is fairness, which is to say that given demands at the inputs, we want to make sure that each queue at the input is scheduled fairly for some definition of fairness. Now, defining fairness is tricky. And there are multiple possible definitions of fairness. Here, we'll look at an important fairness definition called max min fairness.

09 - Max Min Fairness  
  
Now, to define max-min fairness, let's first assume that we have some allocation of rates across flows x i. Now, we say that this allocation is max-min fair if increasing any rate x i implies that some other x j that is smaller than x i must be decreased. To accommodate for the increase in x i. So in other words, the allocation is max-min fair if we can't make, any one of these flow rates better off, without making some flow rate worse off, that's already worse than the flow rate xi. So the upshot results in small demands getting exactly what they asked for, and the larger demands splitting the remaining capacity among themselves equally. More formally, we perform this procedure as follows. We allocate resources to users in order of increasing demand. No user receives more than what they requested. And users that still have unsatisfied demands, split the remaining resources.

10 - Example Max-Min Fairness  
  
Let's consider an example for max-min fair allocation. Let's suppose that we have a link with capacity ten and four demands. 2, 2.6, 4, and 5. Now, obviously the demands exceed capacity. So we need to figure out a way of allocating rates to each of these demands that is max-min fair. First, note that 10 divided by 4 is 2.5. But this is not a good solution, because the first user only needs 2. So the first user would have an excess of .5 under this allocation. So what we want to do is take this excess of .5 and divide it among the remaining 3 users, whose demands have not yet been fulfilled. This would yield an allocation of 2, 2.67, 2.67 and 2.67. But now user 2 has an excess of 0.07, so we take that excess, divide it among the remaining 2, and that gives us our final maximum fair allocation. Note that this is called maximum fairness, because it maximizes the minimum share to each user whose demand is not fully serviced. In this case, the 3rd and 4th users.

11 - Max Min Fairness Quiz  
  
As a quick quiz, let's try doing a max min fair allocation. Suppose that we have demands of one, two, five, and ten, and link, whose rate is 20. Please give the maximum allocation ,across these four users.

12 - Max Min Fairness Solution  
  
To compute the maximum fare allocation, we take 20 and we divide it by 4, which yields 5. But the first user only needs one, which yields an excess of four. The second user only needs two, which yields an excess of three. So in this case the maximum fare allocation is easy. We simply take this excess of seven and give it to the only user whose demand is not yet satisfied, resulting in the maximum and fair allocation of one, two, five, and 12.

13 - How to Achieve Max Min Fairness  
  
Now, how to do we achieve max-min fairness? One approach is via round robin scheduling. Where, given a set of cues, the router simply services them in order. The problem here is that packets may have different sizes. So if the first queue had a huge packet, and the second queue had a little packet, and the third queue had a medium sized packet, then servicing these queues in order obviously isn't fair. Because the first queue would effectively get more of its fair share, because its packet just happened to be bigger. An alternative is to use bit by bit scheduling, where during each time slot, each queue only has one bit serviced. This, of course, is perfectly fair, but the problem is feasibility. How do we service one bit from a queue? A third alternative is called Fair Queuing, which achieves max-min fairness by servicing packets according to the soonest finishing time. A Fair Queuing algorithm computes the virtual finishing time of all candidate packets, which are the packets at the head of all non-empty flow queues. Based on these virtual finishing times, Fair Queuing compares the finishing times of each queue and services the queue with the minimum finishing time. So the queue whose packet has the minimum virtual finishing time is serviced.