An Introduction to Computer Networks

Principle: Packet Switching

Phil: When the Internet was designed, it was based on a controversial and revolutionary idea: packet switching. Nowadays it seems straightforward and the obvious way to build networks. But that wasn't always the case. It's a very simple idea, but of course as it is with simple ideas, there are many interesting implications that arise once you put it into practice. We'll spend an entire week of the course on packet switching and its implications, but in this video we present the high-level idea and its immediate benefits.

What is packet switching?

Packet: A self-contained unit of data that carries information necessary for it to reach its destination.

Packet switching: Independently for each arriving packet, pick its outgoing link. If the link is free, send it. Else hold the packet for later.

CS144, Stanford University

Nick

Packet: A self-contained unit of data that carries information necessary for it to reach its destination.

Packet switching is the idea that we break our data up into discrete, self-contained chunks of data. Each chunk, called a packet, carries sufficient information that a network can deliver the packet to its destination. So let's say we have a source and a destination, and a network of packet switches A, B, and C between them. When A receives a packet for the destination, it sends it along the link to B. When B receives a packet for the destination, it sends it along to C. When C receives a packet for the destination, it sends it to the destination. In the simplest form of packet switching, each

packet is routed separately and independently. For example, let's say there's another switch connected to B, called D. Immediately after sending a packet to C, B can send the next packet to D. Or, if the next packet were also to the destination, it would send two packets back-to-back to C.

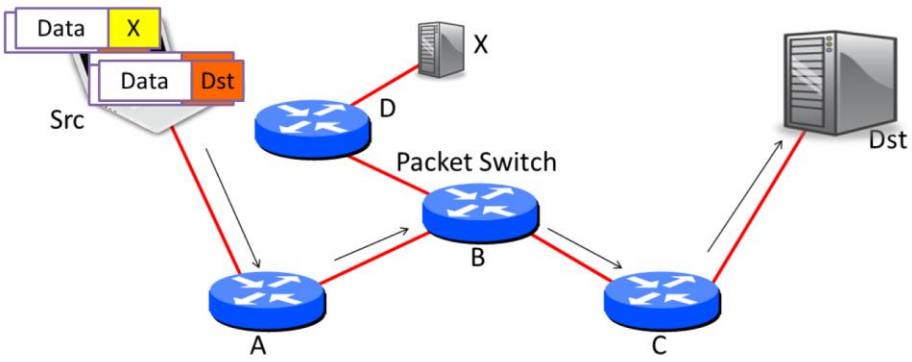
Packet switching: Independently for each arriving packet, pick its outgoing link. If the link is free, send it. Else hold the packet for later.

Here's one example of how packet switching can work: each packet contains an explicit route, specifying the IDs of each packet switch along the way. We call this "self routing" or "source routing," because the source specifies the route. When the source sends a packet, it puts in the packet A, B, C, destination. It then forwards the packet to A. A looks inside the header and sees the next hop is B. So it forwards the packet to B. B sees the next hop is C, and C sees the last hop is the destination. It turns out the Internet supports source routing, but it's generally turned off because it raises big security issues. People owning routers don't want you telling them how to send packets, because maybe you can trick them to sending them somewhere they shouldn't, such as secure computers.

One simple optimization, and what the Internet mostly does today, is to place a small amount of state in each switch which tells it which next hop to send packets to. For example, a switch can have a table of destination addresses

and the next hop. When it receives a packet, it looks up the address in the table, and sends the packet to the appropriate next hop. In this model, all the packet needs to carry is the destination address. Using the address, each switch along the way can make the right decision. For example, in our network here, A's table says that packets to destination should go to switch B, switch B's table says packets to destination should go to switch C, and so on.

Packet Switching

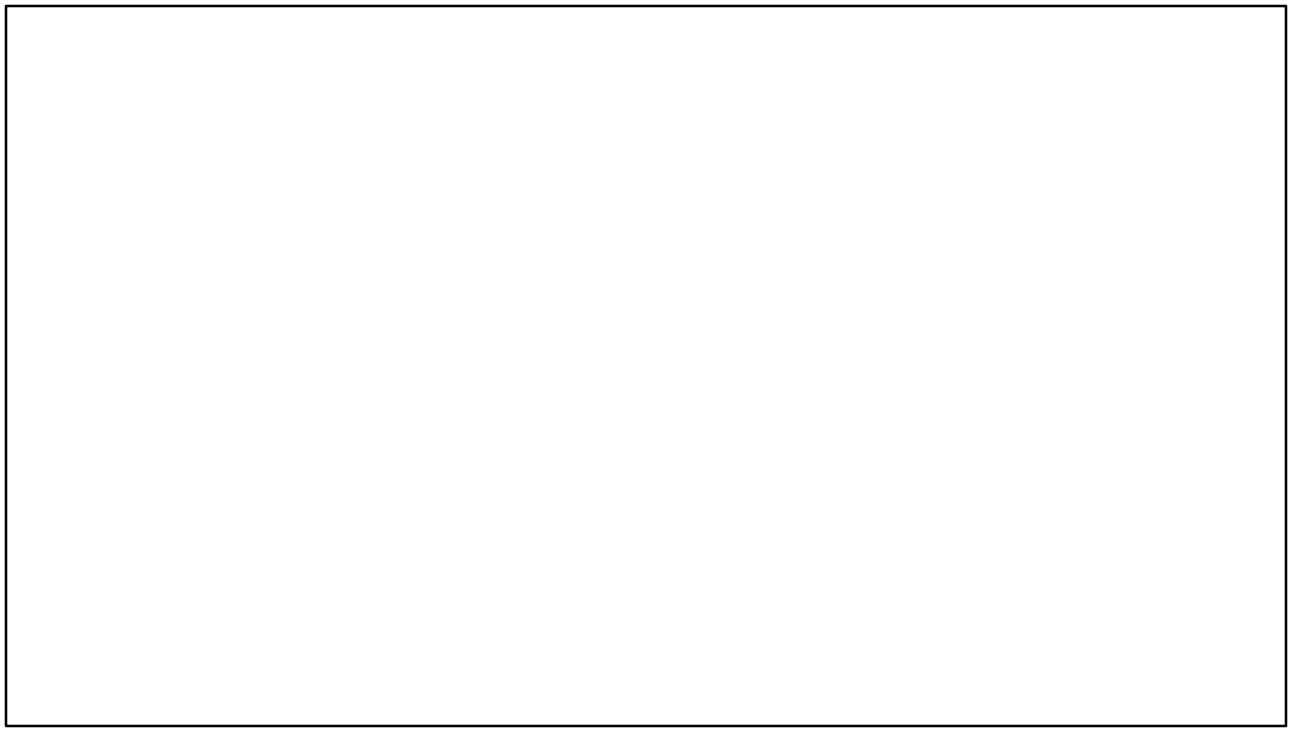


3

In packet switching, there is no dedicated circuit to carry our data. Instead, we send a block of data by adding a header to it, and call it a packet. The header contains the address of where the packet is going, just like an envelope tells the post office where to send a letter.

<click to send packet on link> A packet switched network consists of end-hosts, links, and packet switches. When we send a packet, it is routed hop-by-hop to its destination. Each packet switch lookups the address in the packet header in its local forwarding table.

For example, this packet is addressed to B. When we transmit it, the first router looks up address B in its local table, and sees that switch S2 is the next hop. S2 and S4 do the same thing, and the packet is eventually delivered to B.In the Internet there are several different types of packet switches. Some of them are called routers or gateways, while others are called Ethernet switches. We'll learn more about each of them later. At this stage you just need to know that they are both types of packet switch, and they forward packets based on the destination address in the header.



<two sketches go here:
A: Source routing picture – ABC-Destination
B: Forwarding tables in each switch along the path.>

Two consequences

1. Simple packet forwarding.
2. Efficient sharing of links.

CS144, Stanford University

Phil

(This part seems really based in circuit-based networks. Many students will have never used one. Why would you require per-flow state? Why might you not be able to efficiently share links?)

Packet switching has two really nice properties. The first is that a switch can make individual, local decisions for each packet. It doesn't need to keep extra state on the packets its seen or whether two packets go to the same destination. Even if many packets are part of some larger transfer or protocol, the switch doesn't need to know or care. The switch doesn't need to know that some packets are a Skype call, others are a web request, and others still are a firmware update for your computer. It just forwards packets. This greatly simplifies the switch.

The second is that it lets a switch efficiently share a link between many parties. For example, consider a wireless router in a home with two people browsing the Internet on their laptops. (Draw picture) If one person is reading a page, then the other person can download a file at the full speed of the link. If the first person starts loading a new web page, the link can be shared between the two of them. Once the download completes, the first person can use the full speed of the link.

These two points are really important, so we'll go into some greater detail on both of them.

No per-flow state required

Flow: A collection of datagrams belonging to the same end-to-end communication, e.g. a TCP connection.

Packet switches don't need state for each flow – each packet is self-contained.

No per-flow state to be added/removed.

No per-flow state to be stored.

No per-flow state to be changed upon failure.

CS144, Stanford University

Nick

Of course when we communicate we don't usually send only one packet, we send many; for example a voice call consists of many consecutive packets all part of the same communication. We call this sequence of packets a flow. More specifically:

Flow: A collection of datagrams belonging to the same end-to-end communication, e.g. a TCP connection.

Let's first look at each packet being routed independently.

Because each packet is self-contained, a switch doesn't need to know about groups of packets, or flows of packets. Imagine if every switch had to keep track of every single web connection passing through it. This would require a

huge amount of state that would be hard to manage! Instead, treating each packet independently means the switch can be much simpler to build, manage, and troubleshoot.

The switch doesn't need to worry about adding or removing this per-flow state. Imagine if every time you wanted to load a web page, you had to communicate with every switch along the path to set up state so your request would work. This could make things much slower. Instead, you can just send packets and the switches forward them appropriately.

The switches also don't need to *store* this state. Because switches have to be fast, they'd need to store this state in very fast memory, which is expensive. This lets switches focus on doing one thing, forwarding packets quickly.

Finally, it means switches don't have to worry about failures. Imagine, for example, what happens when you start a web request but then your tablet runs out of energy. The switch is going to keep the per-flow state for the request, but if one of the nodes that created the state fails, the switch needs to know how to clean up after it. Otherwise you can have millions, billions of dead flows eating up your memory. With packet switching, a switch has no per-endpoint state. If your tablet dies, the switch doesn't care, it just means that it stops receiving packets from it. In this way the switch is more functionally independent of the computers sending traffic through it.

Efficient sharing of links

Data traffic is bursty

- Packet switching allows flows to use all available link capacity.
- Packet switching allows flows to share link capacity.

This is called *Statistical Multiplexing*.

CS144, Stanford University

Phil

Think about how you typically use the Internet – your use is bursty. You load a web page, then read it, then load another one. You download a few songs from iTunes, then listen to them. You stream a show from Netflix for forty five minutes, then stop. Data traffic is bursty: rather than always sending and receiving data at a fixed rate, usage jumps and drops, goes up and down, over time.

While there are large-scale changes and peaks in data traffic – 3PM is typically high, as is 8PM, while 2AM is low, on a smaller scale it is very bursty and these bursts are often independent. Let's say you and your friend are both browsing the web in a coffee shop. When you load a new page and when your friend loads a new page are mostly independent. Sometimes they might overlap, but often they

won’t. By treating all of your traffic as just packets, the wireless router can very effectively and simply share its capacity between you. If you’re loading a page while your friend is reading, the wireless router can give all of its capacity to your packets. Similarly, if your friend is loading a page and you’re reading, the router can give all of its capacity to your friend’s packets. The link doesn’t need to go partially idle because one of you isn’t using it, and if you’re both using it then the link can be shared between you.

This idea of taking a single resource and sharing it across multiple users in a probabilistic or statistical way is called statistical multiplexing. It’s statistical in that each user receives a statistical share of the resource based on how much others are using it. For example, if your friend is reading, you can use all of the link. If both of you are loading a page, you receive half of the link capacity.

Summary

Packet switches are simple: they forward packets independently, and don't need to know about flows.

Packet switching is efficient: It lets us efficiently share the capacity among many flows sharing a link.

Nick:

So those are the two major benefits of packet switching: it makes the switches simple because they don't need to know about flows of packets. And second, it lets us efficiently share the capacity among many flows sharing a link. This simple building block was revolutionary at the time, but it's now accepted as the common way to build networks.