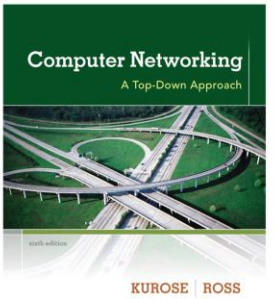
CS320 Lecture 1-2 Introduction



Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
March 2012

The book for this lecture course is shown on the right. There should be >10 copies available in the library

Lectures 1-2 relate to Chapter 1 in book. So you should read pages 27-93.

Labs start next Monday. The first lab is not graded but marks are given for al labs after that. Attendance is mandatory.

Exam = 60%
Labs = 40%

Lectures 1-2: Introduction

our goal:

- get "feel" and terminology
- more depth, detail *later* in course
- approach:
  - use Internet as example

overview:

- what's the Internet?
- what's a protocol?
- network edge; hosts, access net, physical media
- network core: packet/circuit switching, Internet structure
- performance: loss, delay, throughput
- security
- protocol layers, service models
- history

Introduction 1-2

Goal:

This first two lectures (Chapter 1 in the book) presents a broad overview of computer networking and the Internet. Our goal here is to paint a broad picture and set the context for the rest of this course, to see the forest through the trees. We'll cover a lot of ground in this introductory chapter and discuss a lot of the pieces of a computer network, without losing sight of the big picture.

Overview:

>First we'll introduce some basic terminology and then we'll examine the basic hardware and software components that make up a network.

>We'll begin at the network's edge and look at the end systems and network applications running in the network.

>We'll then explore the core of a computer network, examining the links and the switches that transport data, as well as the access networks and physical

media that connect end systems to the network core. We'll learn that the Internet is a network of networks, and we'll learn how these networks connect with

each other.

>After having completed this overview of the edge and core of a computer network, we'll take the broader and more abstract view in the second lecture.

We'll examine delay, loss, and throughput of data in a computer network and provide simple quantitative models for end-to-end throughput and delay: models that take into account transmission, propagation, and queuing delays.

>We'll then introduce some of the key architectural principles in computer networking, namely, protocol layering and service models.

>We'll also learn that computer networks are vulnerable to many different types of attacks; we'll survey some of these attacks and consider how computer networks can be made more secure.

>Finally, we'll close the second lecture with a brief history of computer networking.

Chapter 1: roadmap

1.1 what *is* the Internet? what is a protocol?

1.2 network edge

- end systems, access networks, links

1.3 network core

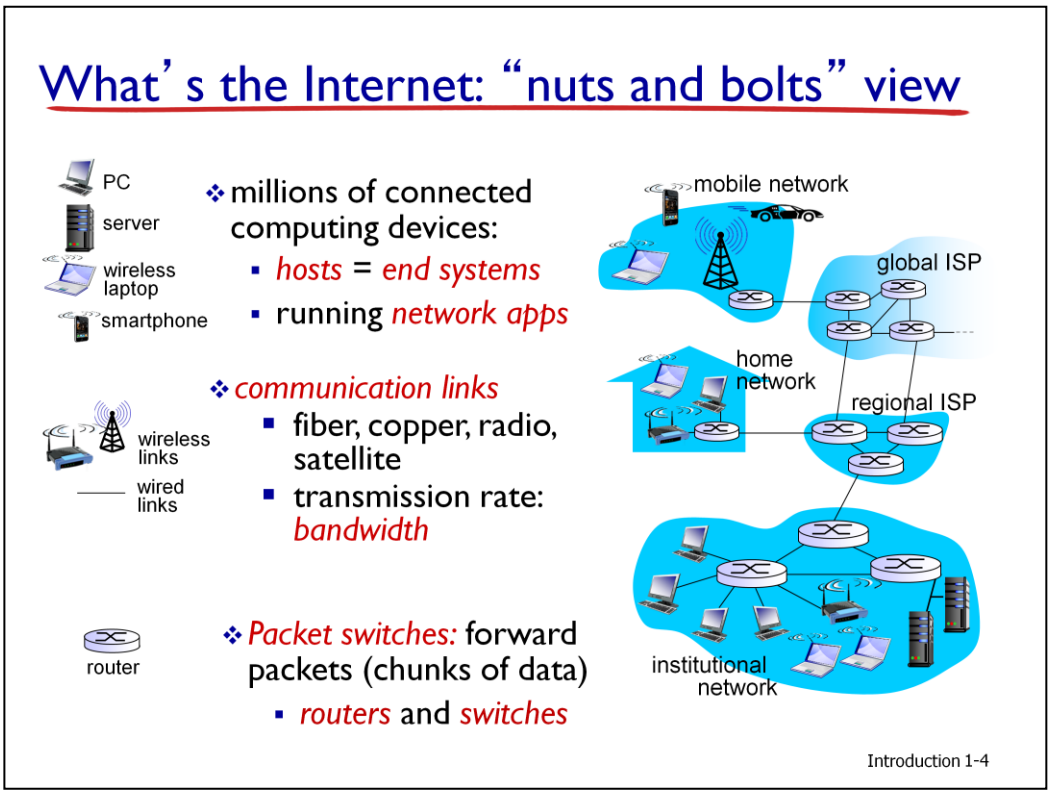
- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

1.6 networks under attack: security

1.7 history



In this course, we'll use the public Internet, a specific computer network, as our principal vehicle for discussing computer networks and their protocols. But what *is* the

Internet? There are a couple of ways to answer this question.

>First, we can describe the nuts and bolts of the Internet, that is, the basic hardware and software components that make up the Internet.

>Second, we can describe the Internet in terms of a networking infrastructure that provides services to distributed applications. (talk about this in a moment)

GO TO NEXT SLIDE

>End systems are connected together by a network of **communication links** and **packet switches**. There are many types of communication links, which are made up of different types of physical media, including coaxial cable, copper wire, optical fiber, and radio spectrum.

>Different links can transmit data at different rates, with the **transmission rate** of a link measured in bits/second.

>When one end system has data to send to another end system, the sending end system segments the data and adds header bytes to each segment. The resulting packages

of information, known as **packets** in the jargon of computer networks, are then sent through the network to the destination end system, where they are reassembled into the original data.

>A packet switch takes a packet arriving on one of its incoming communication links and forwards that packet on one of its outgoing communication links. Packet switches come in many shapes and flavors, but the two most prominent types in today's Internet are **routers** and **link-layer switches**. Both types of switches forward packets toward their ultimate destinations. Link-layer switches are typically used in access networks, while routers are typically used in the network core. The sequence of communication links and packet switches traversed by a packet from the sending end system to the receiving end system is known as a **route** or **path** through the network.

ANOLOGY FACTORY TRUCKS CARGO

"Fun" internet appliances

IP picture frame
http://www.ceiva.com/

Web-enabled toaster +
weather forecaster

Tweet-a-watt:
monitor energy use

Internet
refrigerator

Slingbox: watch,
control cable TV remotely

Internet phones

Introduction 1-5

The Internet is a computer network that interconnects hundreds of millions of computing devices throughout the world. Not too long ago, these computing devices were

primarily traditional desktop PCs, Linux workstations, and so-called servers that store and transmit information such as Web pages and e-mail messages. Increasingly,

however, nontraditional Internet end systems such as laptops, smartphones, tablets, TVs, gaming consoles, Web cams, automobiles, environmental sensing devices,

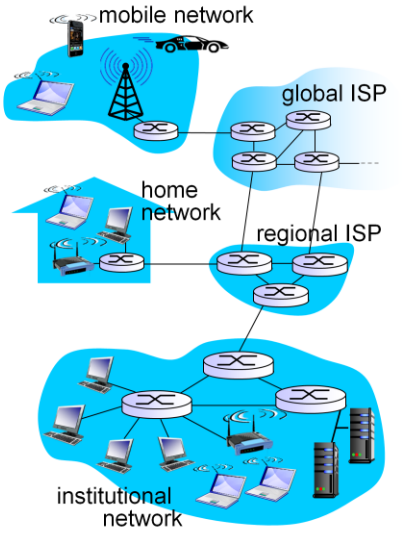
picture frames, and home electrical and security systems are being connected to the Internet. Indeed, the term *computer network* is beginning to sound a bit dated, given

the many nontraditional devices that are being hooked up to the Internet. In Internet jargon, all of these devices are called **hosts** or **end systems**.

Currently 8.4 billion connected things. 3.6 billion internet users (7.5 billion people in the world)

What's the Internet: "nuts and bolts" view

- *Internet:* "network of networks"
  - Interconnected ISPs
- *protocols* rules that control sending, receiving of msgs
  - e.g., TCP, IP, HTTP, Skype, 802.11
- *Internet standards*
  - RFC: Request for comments
  - IETF: Internet Engineering Task Force



Introduction 1-6

>End systems access the Internet through **Internet Service Providers (ISPs)**, including residential ISPs such as local cable or telephone companies; corporate ISPs; university ISPs; Each ISP is in itself a network of packet switches and communication links.

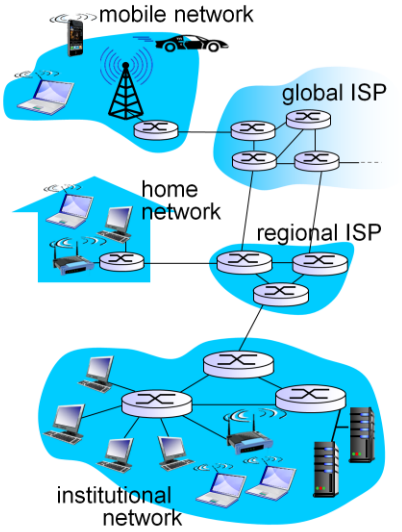
>An upper-tier ISP consists of high-speed routers interconnected with high-speed fiber-optic links. Each ISP network, whether upper-tier or lower-tier, is managed independently, runs the IP **protocol** (more later), and conforms to certain naming and address conventions. We'll examine ISPs and their interconnection more closely later.

>End systems, packet switches, and other pieces of the Internet run **protocols** that control the sending and receiving of information within the Internet. The **Transmission Control Protocol (TCP)** and the **Internet Protocol (IP)** are two of the most important protocols in the Internet. The IP protocol specifies the format of the packets that are sent and received among routers and end systems.

>Given the importance of protocols to the inet, it's important that everyone agree on what each and every protocol does, so that people can create systems and products that interoperate. This is where standards come into play. **Internet standards** are developed by the Internet Engineering Task Force (IETF). The IETF standards documents are called **requests for comments (RFCs)**. RFCs tend to be quite technical and detailed-define protocols such as TCP, IP, HTTP (for the Web), and SMTP (for e-mail). currently>6,000 RFCs.

What's the Internet: a service view

- *Infrastructure that provides services to applications:*
  - Web, VoIP, email, games, e-commerce, social nets, …
- *provides application programming interface (API) to apps*
  - hooks that allow sending and receiving app programs to "connect" to Internet
  - provides service options, analogous to postal service



Introduction 1-7

But what *is* the Internet? We said there are a couple of ways to answer this question.

>Second, we can describe the Internet in terms of a networking infrastructure that provides services to distributed applications.

These applications include electronic mail, Web surfing, social networks, instant messaging, Voiceover- IP (VoIP), video streaming, distributed games, peer-to-peer (P2P) file sharing, television over the Internet, etc etc. The applications are said to be **distributed applications**, since they involve multiple end systems that exchange data with each other.

Importantly, Internet applications run on end systems—they do not run in the packet switches in the network core. Although packet switches facilitate the exchange of data among end systems, they are not concerned with the application that is the source or sink of data.

End systems attached to the Internet provide an **Application Programming Interface (API)** that specifies how a program running on one end system asks the Internet infrastructure to deliver data to a specific destination program running on another end system. This Internet API is a set of rules that the sending

program must follow so that the Internet can deliver the data to the destination program. We'll discuss the Internet API in detail next week.

ANOLOGY ALICE POSTS LETTER TO BOB

The postal service, of course, provides more than one service to its customers. It provides express delivery, reception confirmation, ordinary use, and many more

services. In a similar manner, the Internet provides multiple services to its applications. When you develop an Internet application, you too must choose one of the

Internet's services for your application. We'll describe the Internet's services next week.

What's a protocol?

human protocols:

- "what's the time?"
- "I have a question"
- introductions

… specific msgs sent

… specific actions taken when msgs received, or other events

network protocols:

- machines rather than humans
- all communication activity in Internet governed by protocols

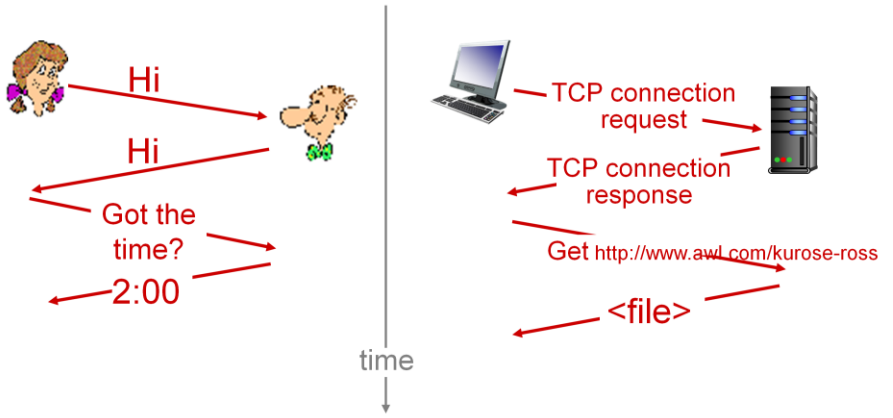
protocols define format, order of msgs sent and received among network entities, and actions taken on msg transmission, receipt

It is probably easiest to understand the notion of a computer network protocol by first considering some human analogies, since we humans execute protocols (good manners!) all of the time. Consider what you do when you want to ask someone for the time of day.

JUST READ SLIDE

What's a protocol?

a human protocol and a computer network protocol:



Q: other human protocols?

Introduction 1-9

A typical exchange is shown in the figure on the left. Human protocol (or good manners, at least) dictates that one first offer a greeting (the first "Hi") to initiate communication with someone else. The typical response to a "Hi" is a returned "Hi" message. Implicitly, one then takes a cordial "Hi" response as an indication that one can proceed and ask for the time of day. A different response to the initial "Hi" (such as "Don't bother me!" or "I don't speak English," or some unprintable reply) might indicate an unwillingness or inability to communicate. In this case, the human protocol would be not to ask for the time of day. Sometimes one gets no response at all to a question, in which case one typically gives up asking that person for the time. Note that in our human protocol, *there are specific messages we send, and specific actions we take in response to the received reply messages or other events* (such as no reply within some given amount of time).

>As an example of a computer network protocol with which you are probably familiar, consider what happens when you make a request to a Web server, that is, when you type the URL of a Web page into your Web browser. The scenario is illustrated in the right half of Figure 1.2. First, your computer will send a connection request message to the Web server and wait for a reply. The Web server will eventually receive your connection request message and return a connection reply message. Knowing that it is now OK to request the Web document, your computer then sends the name of the Web page it wants to fetch from that Web server in a GET message. Finally, the Web server returns the Web page (file) to your computer.

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

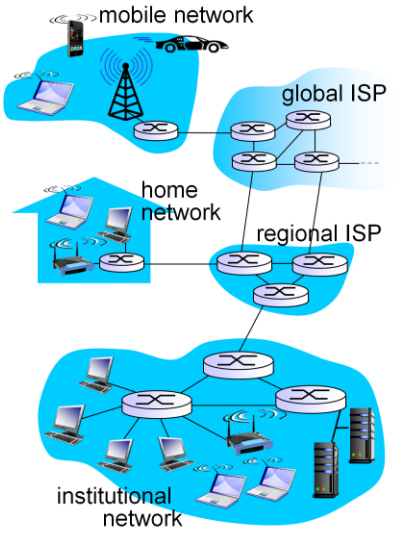
1.6 networks under attack: security

1.7 history

We have just discussed a high-level overview of the Internet and networking protocols. We are now going to delve a bit more deeply into the components of a computer network (and the Internet, in particular). We begin in this section at the edge of a network and look at the components with which we are most familiar—namely, the computers, smartphones and other devices that we use on a daily basis. In the next section we'll move from the network edge to the network core and examine switching and routing in computer networks.

A closer look at network structure:

- *network edge:*
  - hosts: clients and servers
  - servers often in data centers
- *access networks, physical media:* wired, wireless communication links
- *network core:*
  - interconnected routers
  - network of networks



Introduction 1-11

>Recall from the previous section that in computer networking jargon, the computers and other devices connected to the Internet are often referred to as end systems or hosts.

They are referred to as end systems because they sit at the edge of the Internet (see figure). End systems are also referred to as *hosts* because they host (that is, run) application programs such as a Web browser program, a Web server program, an e-mail etc.

>Hosts are sometimes further divided into two categories: **clients** and **servers**. Informally, clients tend to be desktop and mobile PCs, smartphones, and so on, whereas servers tend to be more powerful machines that store and distribute Web pages, stream video, relay e-mail, and so on. Today, most of the servers from which we receive search results, web pages etc are in large data centres. Google has >50 data centers, with many having more than one 100k servers.

>Having considered the applications and end systems at the "edge of the network," let's next consider the access network—the network that physically connects an end system to the first router (also known as the "edge router") on a path from the end system to any other distant end system. These communication links can be wired or wireless and well discuss access networks a little more on the next 5 slides.

>When we have discussed the access netowrk we'll move further inwards to discuss the network core. LOOK AT SLIDE.

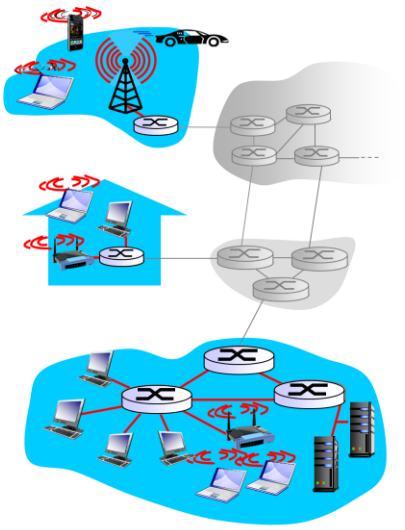
Access networks and physical media

Q: How to connect end systems to edge router?

- residential access nets
- institutional access networks (school, company)
- mobile access networks

keep in mind:

- bandwidth (bits per second) of access network?
- shared or dedicated?



Introduction 1-12

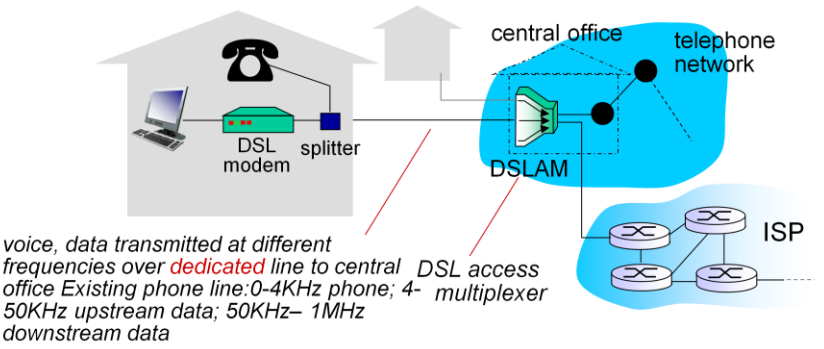
The access network is the network that physically connects an end system to the first router (also known as the "edge router"). The figure on the side shows several types of access networks with thick, red lines, and the settings (home, enterprise/institutional, and wide-area mobile wireless) in which they are used.

DISCUSS SLIDE

Today, the two most prevalent types of broadband residential access are **digital subscriber line (DSL)** and cable. A residence typically obtains DSL Internet access

from the same local telephone company (telco) that provides its wired local phone access. Thus, when DSL is used, a customer's telephone company is also its ISP.

Access net: digital subscriber line (DSL)



- use *existing* telephone line to central office DSLAM
  - data over DSL phone line goes to Internet
  - voice over DSL phone line goes to telephone net
- < 2.5 Mbps upstream transmission rate (typically < 1 Mbps)
- < 24 Mbps downstream transmission rate (typically < 10 Mbps)

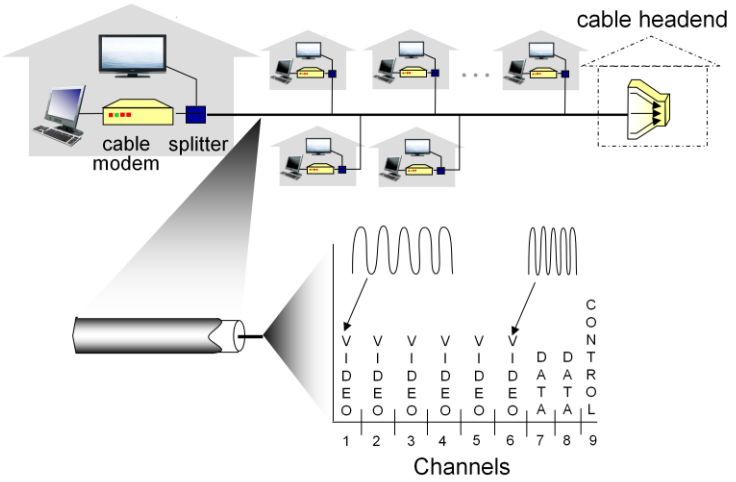
Introduction 1-13

As shown in this figure, each customer's DSL modem uses the existing telephone line (twisted pair copper wire, which we'll discuss shortly) to exchange data with a **digital subscriber line access multiplexer (DSLAM)** located in the telephone company's local central office (CO). The home's DSL modem takes digital data and translates it to high frequency tones for transmission over phone wires to the CO; the analog signals from many houses are translated back into digital format at the DSLAM. The residential telephone line carries both data and traditional telephone signals simultaneously, which are encoded at different frequencies REFER TO FIGURE FOR FREQs:

This approach makes the single DSL link appear as if there were three separate links, so that a telephone call and an Internet connection can share the DSL link at the same time. (We'll describe this technique of frequency-division multiplexing shortly). On the customer side, a splitter separates the data and telephone signals arriving to the home and forwards the data signal to the DSL modem. On the telephone companies side, in the CO, the DSLAM separates the data and phone signals and sends the data into the Internet. Hundreds or even thousands of households connect to a single DSLAM.

>The DSL standards define transmission rates of 12 Mbps downstream and 1.8 Mbps upstream, and 24 Mbps downstream and 2.5 Mbps upstream.

Access net: cable network



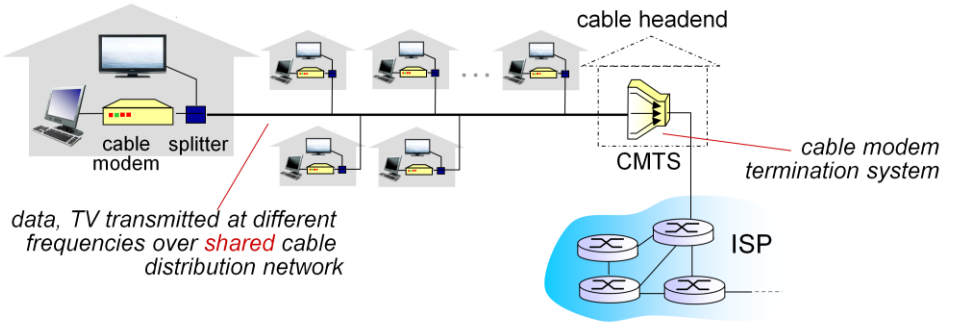
frequency division multiplexing: different channels transmitted in different frequency bands

Introduction 1-14

While DSL makes use of the existing local telephone infrastructure, **cable Internet access** makes use of the cable television company's existing cable television infrastructure. A residence obtains cable Internet access from the same company that provides its cable television. As illustrated in the figure, fiber optics connect the cable head end to neighborhood-level junctions, from which traditional coaxial cable is then used to reach individual houses and apartments. Each neighborhood junction typically supports 500 to 5,000 homes. Because both fiber and coaxial cable are employed in this system, it is often referred to as hybrid fiber coax (HFC).

DISCUSS FREQUENCY DIVISION MULTIPLEXING

Access net: cable network



- HFC: hybrid fiber coax
  - asymmetric: up to 30Mbps downstream transmission rate, 2 Mbps upstream transmission rate
- network of cable, fiber attaches homes to ISP router
  - homes *share access network* to cable headend
  - unlike DSL, which has dedicated access to central office

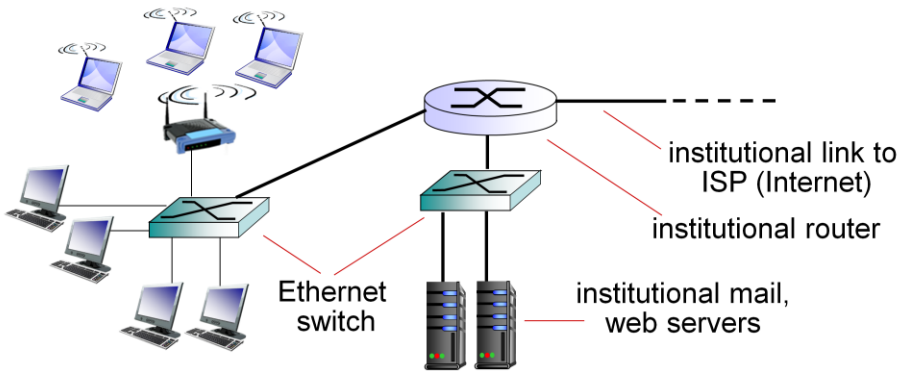
Introduction 1-15

Cable internet access requires special modems, called cable modems. As with a DSL modem, the cable modem is typically an external device and connects to the home PC through an Ethernet port. (We will discuss Ethernet in great detail in later in the course) At the cable head end, the cable modem termination system (CMTS) serves a similar function as the DSL network's DSLAM—turning the analog signal sent from the cable modems in many downstream homes back into digital format. Cable modems divide the HFC network into two channels, a downstream and an upstream channel. As with DSL, access is typically asymmetric, with the downstream channel typically allocated a higher transmission rate than the upstream channel. The DOCSIS 2.0 standard defines downstream rates up to 42.8 Mbps and upstream rates of up to 30.7 Mbps. As in the case of DSL networks, the maximum achievable rate may not be realized due to lower contracted data rates or media impairments.

Cable has shared access – contention among resisdence. A distributed multiple access protocol is needed to coordinate transmissions and avoid collisions. We'll chat about his later in the course.

Other hybrids have emerged including FTTH as well as a single fiber to a number of nearby homes.

Enterprise access networks (Ethernet)



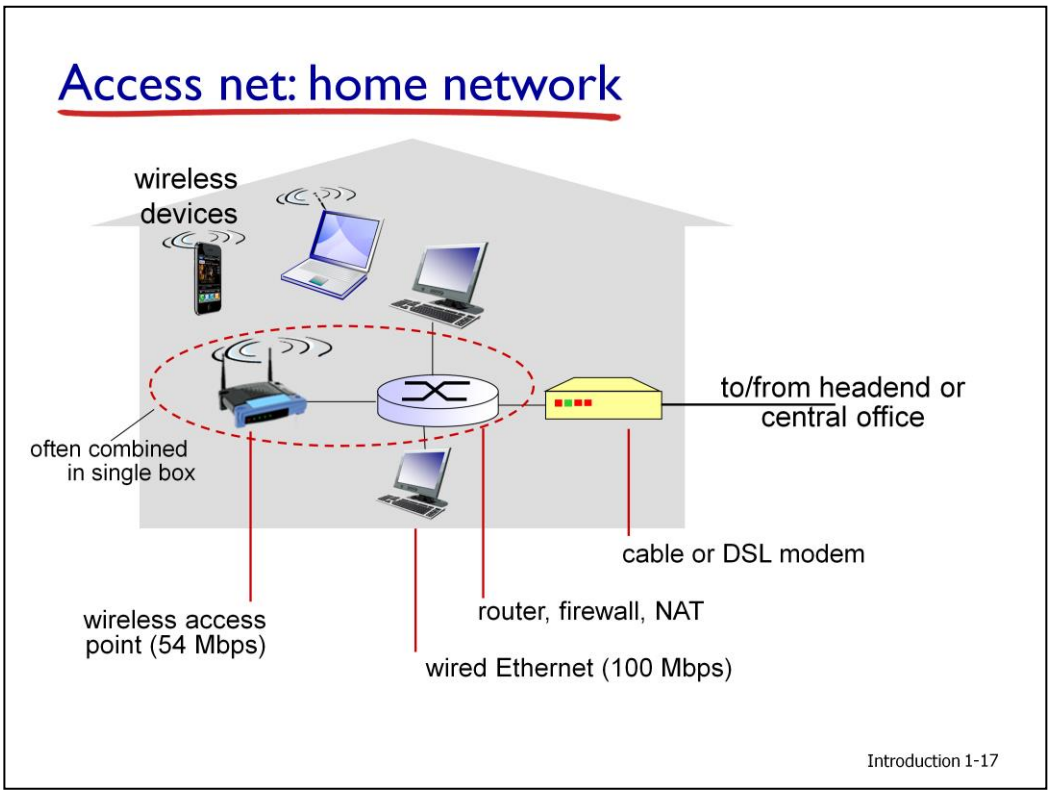
- typically used in companies, universities, etc
- 10 Mbps, 100Mbps, 1Gbps, 10Gbps transmission rates
- today, end systems typically connect into Ethernet switch

Introduction 1-16

On corporate and university campuses, and increasingly in home settings, a local area network (LAN) is used to connect an end system to the edge router. Although there are many types of LAN technologies, Ethernet is by far the most prevalent access technology in corporate, university, and home networks. As shown in the figure, Ethernet users use twisted-pair copper wire to connect to an Ethernet switch (we'll discuss ethernet much more later in the course). The Ethernet switch, or a network of such interconnected switches, is then in turn connected into the larger Internet. With Ethernet access, users typically have 100 Mbps access to the Ethernet switch, whereas servers may have 1 Gbps or even 10 Gbps access.

>Increasingly, however, people are accessing the Internet wirelessly from laptops, smartphones, tablets, and other devices. In a wireless LAN setting, wireless users transmit/receive packets to/from an access point that is connected into the enterprise's network (most likely including wired Ethernet), which in turn is connected to the wired Internet. A wireless LAN user must typically be within a few tens of meters of the access point. Wireless LAN access based on IEEE 802.11 technology, more colloquially known

as WiFi, is now just about everywhere—universities, business offices, cafes, airports, homes, and even in airplanes. 802.11 today provides a shared transmission rate of up to 54 Mbps - We might cover WiFi later in the couse…to be decided.



Even though Ethernet and WiFi access networks were initially deployed in enterprise (corporate, university) settings, they have recently become relatively common components of home networks. Many homes combine broadband residential access (that is, cable modems or DSL) with these inexpensive wireless LAN technologies to create powerful home networks. This slide shows a typical home network. This home network consists of a smartphone, a roaming laptop as well as 2 wired PCs; a base station (the wireless access point), which communicates with the wireless PC; a cable modem, providing broadband access to the Internet; and a router, which interconnects the base station and the stationary PC with the cable modem.

Wireless access networks

- shared *wireless* access network connects end system to router
  - via base station aka "access point"

wireless LANs:

- within building (100 ft)
- 802.11b/g (WiFi): 11, 54 Mbps transmission rate

wide-area wireless access

- provided by mobile phone company 10's km
- between 1 and 100 Mbps
- 3G, 4G, 5G

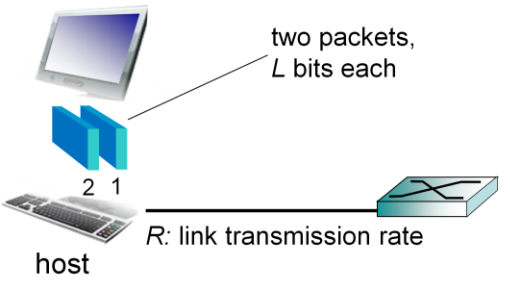
Introduction 1-18

Increasingly, smartphones are being used to send email, surf the Web, Tweet, and download music while on the run. These devices employ the same wireless infrastructure used for mobile telephony to send/receive packets through a base station that is operated by the mobile phone network provider. Unlike WiFi, a user need only be within a few tens of kilometers (as opposed to a few tens of meters) of the base station. Telecommunications companies have made enormous investments in so-called third-and fourth generation (3G-4G) wireless, which provides packet-switched wide-area wireless Internet access at speeds in excess of 10 Mbps. But even higher-speed wide-area access technologies—a fifth-generation (5G) of wide-area wireless networks—are already being researched (1Gbs).

Host: sends *packets* of data

host sending function:

- takes application message
- breaks into smaller chunks, known as *packets*, of length *L* bits
- transmits packet into access network at *transmission rate R*
  - link transmission rate, aka link *capacity, aka link bandwidth*



$$\text{packet transmission delay} = \text{time needed to transmit } L\text{-bit packet into link} = \frac{L \text{ (bits)}}{R \text{ (bits/sec)}}$$

1-19

Time for some gentel methematics

DISCUSS SLIDE

Lets now discuss the physcial media that make up the access network in the context of transmission rate or bits per second

Physical media

- bit: propagates between transmitter/receiver pairs
- physical link: what lies between transmitter & receiver
- guided media:
  - signals propagate in solid media: copper, fiber, coax
- unguided media:
  - signals propagate freely, e.g., radio

twisted pair (TP)

- two insulated copper wires
  - Category 5: 100 Mbps, 1 Gpbs Ethernet
  - Category 6: 10Gbps

Introduction 1-20

In the previous slides, we gave an overview of some of the most important network access technologies in the Internet. As we described these technologies, we also indicated the physical media used. For example, we said that HFC uses a combination of fiber cable and coaxial cable. We said that DSL and Ethernet use copper wire. And we said that mobile access networks use the radio spectrum. In the next few slides we provide a very brief overview of these and other transmission media that are commonly used in the Internet.

In order to define what is meant by a physical medium, let us reflect on the brief life of a **bit**. Consider a bit traveling from one end system, through a series of links and routers, to another end system. This poor bit gets kicked around and transmitted many, many times! The source **end system** first transmits the bit, and shortly thereafter the first router in the series receives the bit; the first router then transmits the bit, and shortly thereafter the second router receives the bit; and so on. Thus our bit, when traveling from source to destination, passes through a series of **transmitter-receiver pairs**. For each transmitter-receiver pair, the bit is sent by propagating electromagnetic waves or optical pulses across a **physical medium**.

The physical medium can take many shapes and forms and does not have to be of the same type for each transmitter-receiver pair along the path.

Examples of physical media include twisted-pair copper wire, coaxial cable, multimode fiber-optic cable, terrestrial radio spectrum, and satellite radio spectrum.

Physical media fall into two categories: **guided media** and **unguided media**. With guided media, the waves are guided along a solid medium, such as a fiber-optic cable, a twisted-pair copper wire, or a coaxial cable. With unguided media, the waves propagate in the atmosphere and in outer space, such as in a wireless LAN or a digital satellite channel.

The least expensive and most commonly used guided transmission medium is twisted-pair copper wire. 100 years. Twisted pair consists of two insulated copper wires, each about 1 mm thick, arranged in a regular spiral pattern. The wires are twisted together to reduce the electrical interference from similar pairs close by. Typically, a number of pairs are bundled together in a cable by wrapping the pairs in a protective shield. A wire pair constitutes a single communication link. **Unshielded twisted pair (UTP)** is commonly used for computer networks within a building, that is, for LANs. Data rates for LANs using twisted pair today range from 10 Mbps to 10 Gbps. The data rates that can be achieved depend on the thickness of the wire and the distance between transmitter and receiver.

Physical media: coax, fiber

coaxial cable:

- two concentric copper conductors
- bidirectional
- broadband:
  - multiple channels on cable
  - HFC

fiber optic cable:

- glass fiber carrying light pulses, each pulse a bit
- high-speed operation:
  - high-speed point-to-point transmission (e.g., 10's-100's Gpbs transmission rate)
- low error rate:
  - repeaters spaced far apart
  - immune to electromagnetic noise

Introduction 1-21

Like twisted pair, coaxial cable consists of two copper conductors, but the two conductors are concentric rather than parallel. With this construction and special insulation and shielding, coaxial cable can achieve high data transmission rates. Coaxial cable is quite common in cable TV. As we saw earlier, cable TV systems have recently been coupled with cable modems to provide homes with Internet access at rates of tens of Mbps. In cable TV and cable Internet access, the transmitter shifts the digital signal to a specific frequency band, and the resulting analog signal is sent from the transmitter to one or more receivers. Coaxial cable can be used as a guided **shared medium**. Specifically, a number of end systems can be connected directly to the cable, with each of the end systems receiving whatever is sent by the other end systems.

>An optical fiber is a thin, flexible medium that guides pulses of light, with each pulse representing a bit. A single optical fiber can support tremendous bit rates, up to tens or even hundreds of gigabits/second. They are immune to electromagnetic interference, have very low signal attenuation up to 100 kilometers, and are very hard to tap. These characteristics have made fiber optics the preferred longhaul guided transmission media, particularly for overseas links. Many of the long distance telephone networks in the US and elsewhere now use fiber optics exclusively. Forms the backbone of the Internet. However, the high cost of optical devices—such as transmitters, receivers, and switches—has hindered their deployment for short-haul transport, such as in a LAN.

Physical media: radio

- signal carried in electromagnetic spectrum
- no physical "wire"
- bidirectional
- propagation environment effects:
  - reflection
  - obstruction by objects
  - interference

radio link types:

- terrestrial microwave
  - e.g. up to 45 Mbps channels
- LAN (e.g., WiFi)
  - 11Mbps, 54 Mbps
- wide-area (e.g., cellular)
  - 3G cellular: ~ few Mbps
- satellite
  - Kbps to 45Mbps channel (or multiple smaller channels)
  - 270 msec end-end delay
  - geosynchronous versus low orbitting altitude

>Radio channels carry signals in the electromagnetic spectrum. They are an attractive medium because they require no physical wire to be installed, can penetrate walls, provide connectivity to a mobile user, and can potentially carry a signal for long distances.

>The characteristics of a radio channel depend significantly on the propagation environment and the distance over which a signal is to be carried. Environmental considerations determine path loss and shadow fading (which decrease the signal strength as the signal travels over a distance and around/through obstructing objects), multipath fading (due to signal reflection off of interfering objects), and interference (due to other transmissions and electromagnetic signals).

>Terrestrial radio channels can be broadly classified into three groups: those that operate over very short distance (e.g., with one or two meters); those that operate in local areas, typically spanning from ten to a few hundred meters; and those that operate in the wide area, spanning tens of kilometers. Personal devices such as wireless headsets, keyboards, and medical devices operate over short distances; home wireless routers use local-area radio channels; the mobile phone access technologies use wide-area radio channels. We'll may discuss radio channels in detail later.

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

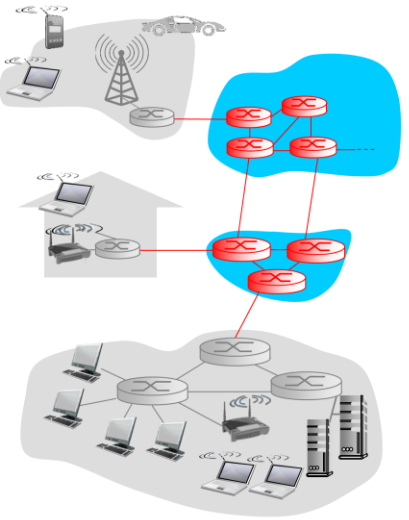
1.6 networks under attack: security

1.7 history

Having examined the Internet's edge, let us now delve more deeply inside the network core—the mesh of packet switches and links that interconnects the Internet's end systems.

The network core

- mesh of interconnected routers
- packet-switching: hosts break application-layer messages into *packets*
  - forward packets from one router to the next, across links on path from source to destination
  - each packet transmitted at full link capacity (recall: takes L/R seconds)



Introduction 1-24

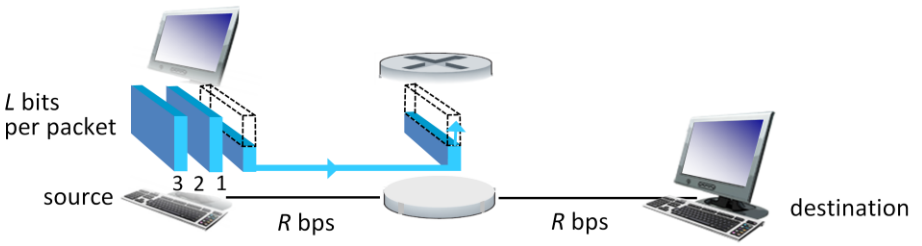
The figure on the right highlights the network core in colour; essentially a mesh of interconnected routers.

In a network application, end systems or hosts exchange **messages** with each other. Messages can contain anything the application designer wants. Messages may perform a control function or contain data, such as an email message, a JPEG image, or an MP3 audio file.

To send a message from a source end system to a destination end system, the source breaks long messages into smaller chunks of data known as **packets**. Between source and destination, each packet travels through communication links and **packet switches** (for which there are two predominant types, **routers** and **linklayer switches**).

Packets are transmitted over each communication link at a rate equal to the *full* transmission rate of the link. So, if a source end system or a packet switch is sending a packet of *L* bits over a link with transmission rate *R* bits/sec, then the time to transmit the packet is *L/R* seconds.

Packet-switching: store-and-forward



- takes *L*/*R* seconds to transmit (push out) *L*-bit packet into link at *R* bps
- *store and forward:* entire packet must arrive at router before it can be transmitted on next link
- end-end delay = 2*L*/*R* (assuming zero propagation delay)

one-hop numerical example:

- *L* = 7.5 Mbits
- *R* = 1.5 Mbps
- one-hop transmission delay = 5 sec

more on delay shortly …

Introduction 1-25

Most packet switches use **store-and-forward transmission** at the inputs to the links. Store-and-forward transmission means that the packet switch must receive the entire packet before it can begin to transmit the first bit of the packet onto the outbound link. To explore store-and-forward transmission in more detail, consider a simple network consisting of two end systems connected by a single router, as shown in this slide. A router will typically have many incident links, since its job is to switch an incoming packet onto an outgoing link; in this simple example, the router has the rather simple task of transferring a packet from one (input) link to the only other attached link. In this example, the source has three packets, each consisting of *L* bits, to send to the destination. At the snapshot of time shown in this slide, the source has transmitted some of packet 1, and the front of packet 1 has already arrived at the router. Because the router employs store-and-forwarding, at this instant of time, the router cannot transmit the bits it has received; instead it must first buffer (i.e., "store") the packet's bits. Only after the router has received *all* of the packet's bits can it begin to transmit (i.e., "forward") the packet onto the outbound link.

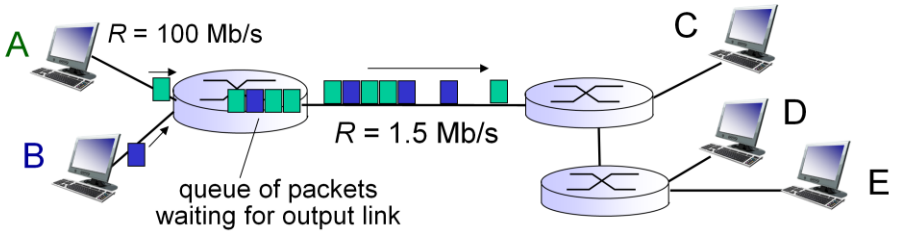
To gain some insight into store-and-forward transmission, let's now calculate the amount of time that elapses from when the source begins to send the packet until the destination has received the entire packet. (Here we will ignore propagation delay—the time it takes for the bits to travel across the wire at near the speed of light—which will be discussed later.) The source

begins to transmit at time 0; at time L/R seconds…….

Now let's calculate the amount of time that elapses from when the source begins to send the first packet until the destination has received all three packets. As before, at time L/R, the router begins to forward the first packet….

What if there were N links?

Packet Switching: queueing delay, loss



queuing and loss:

- If arrival rate (in bits) to link exceeds transmission rate of link for a period of time:
  - packets will queue, wait to be transmitted on link
  - packets can be dropped (lost) if memory (buffer) fills up

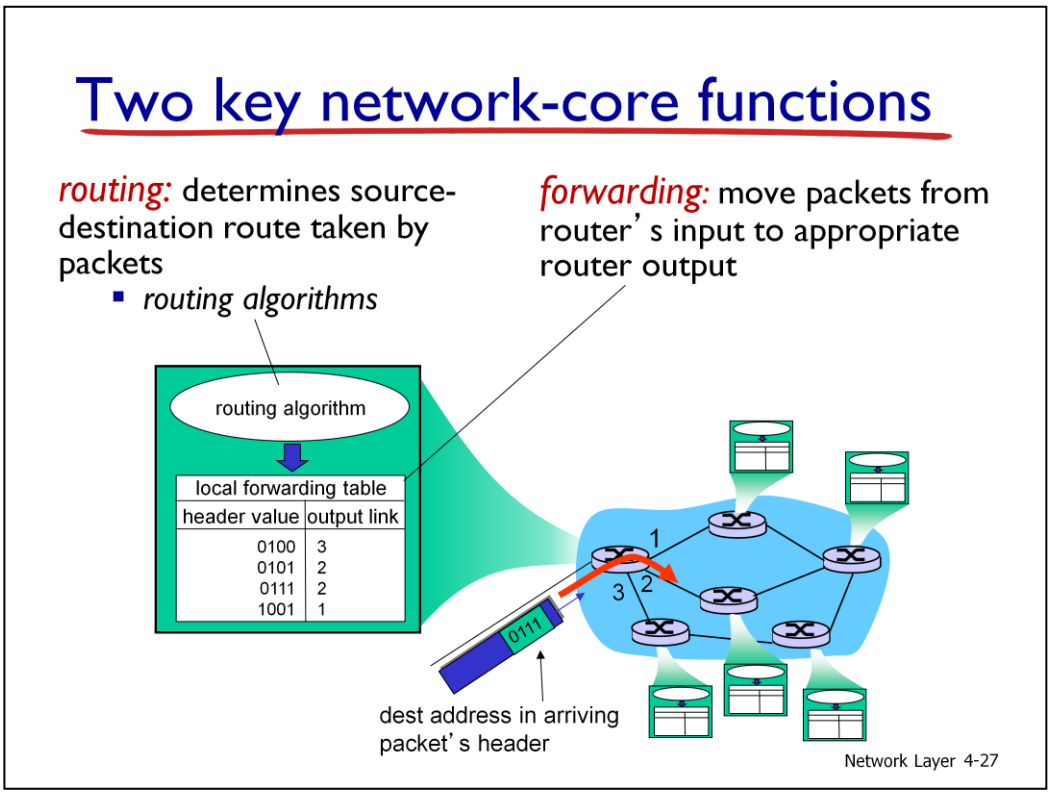
Introduction 1-26

Each packet switch has multiple links attached to it. For each attached link, the packet switch has an **output buffer** (also called an **output queue**), which stores packets that the router is about to send into that link. The output buffers play a key role in packet switching.

If an arriving packet needs to be transmitted onto a link but finds the link busy with the transmission of another packet, the arriving packet must wait in the output buffer. Thus, in addition to the store-and-forward delays, packets suffer output buffer **queuing delays**.

These delays are variable and depend on the level of congestion in the network. Since the amount of buffer space is finite, an arriving packet may find that the buffer is completely full with other packets waiting for transmission. In this case, **packet loss** will occur—either the arriving packet or one of the already-queued packets will be dropped.

We will discuss packet delay more later in the course.



Earlier, we said that a router takes a packet arriving on one of its attached communication links and forwards that packet onto another one of its attached communication links. But how does the router determine which link it should forward the packet onto? Packet forwarding is actually done in different ways in different types of computer networks.

In the Internet, every end system has an address called an IP address. When a source end system wants to send a packet to a destination end system, the source includes the destination's IP address in the packet's header. As with postal addresses, this address has a hierarchical structure. When a packet arrives at a router in the network, the router examines a portion of the packet's destination address and forwards the packet to an adjacent router. More specifically, each router has a **forwarding table** that maps destination addresses (or portions of the destination addresses) to that router's outbound links. When a packet arrives at a router, the router examines the address and searches its forwarding table, using this destination address, to find the appropriate outbound link. The router then directs the packet to this outbound link.

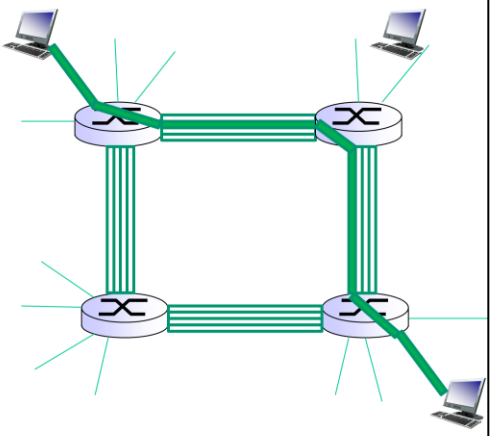
The end-to-end routing process is analogous to a car driver who does not use maps but instead prefers to ask for directions.

How do forwarding tables get set? Are they configured by hand in each and every router, or does the Internet use a more automated procedure? The Internet has a number of special **routing protocols** that are used to automatically set the forwarding tables. A routing protocol may, for example, determine the shortest path from each router to each destination and use the shortest path results to configure the forwarding tables in the routers. We will discuss further later in the course.

Alternative core: circuit switching

end-end resources allocated to, reserved for "call" between source & dest:

- In diagram, each link has four circuits.
  - call gets 2nd circuit in top link and 1st circuit in right link.
- dedicated resources: no sharing
  - circuit-like (guaranteed) performance
- circuit segment idle if not used by call *(no sharing)*
- Commonly used in traditional telephone networks



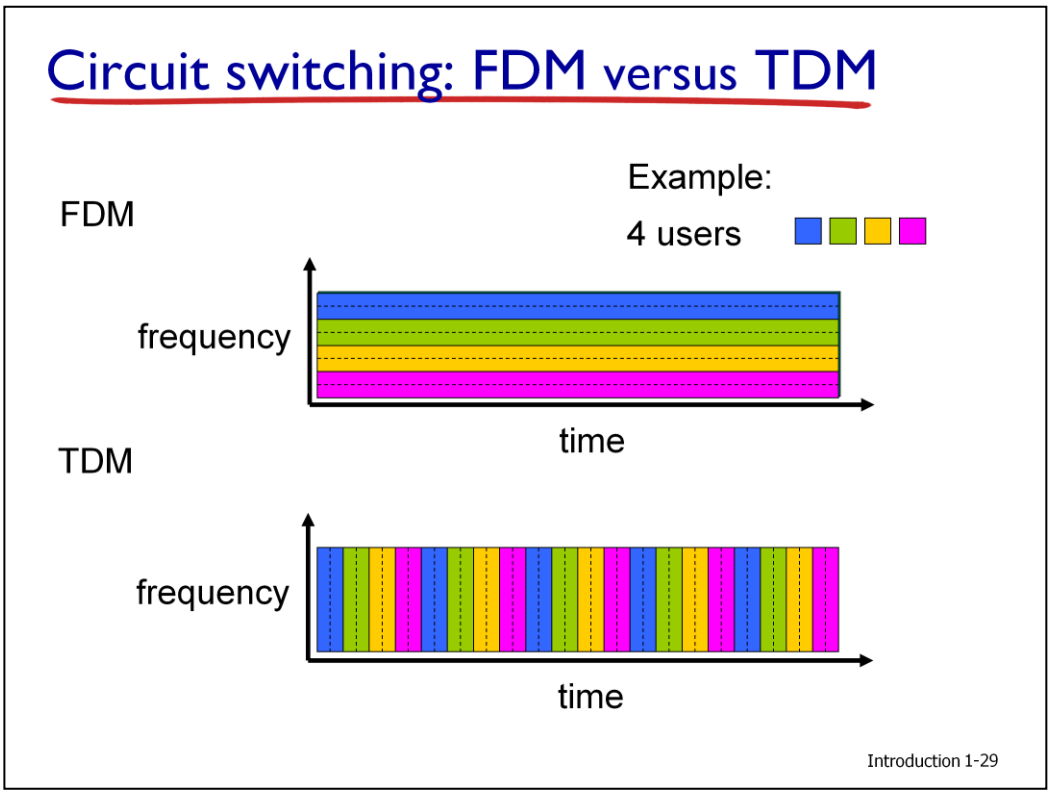
Introduction 1-28

There are two fundamental approaches to moving data through a network of links and switches: **circuit switching** and **packet switching**. Having covered packetswitched networks in the previous subsection, we now turn our attention to circuitswitched networks.

In circuit-switched networks, the resources needed along a path (buffers, link, transmission rate) to provide for communication between the end systems are *reserved* for the duration of the communication session between the end systems. In packet-switched networks, these resources are *not* reserved; a session's messages use the resources on demand, and as a consequence, may have to wait (that is, queue) for access to a communication link.

As a simple analogy, consider two restaurants, one that requires reservations and another that neither requires reservations nor accepts them….

Traditional telephone networks are examples of circuit-switched networks; the network must establish a connection 'circuit' between the sender and the receiver.



A circuit in a link is implemented with either **frequency-division multiplexing (FDM)** or **time-division multiplexing (TDM)**. With FDM, the frequency spectrum of a link is divided up among the connections established across the link.

Specifically, the link dedicates a frequency band to each connection for the duration of the connection. In telephone networks, this frequency band typically has a width of 4 kHz (that is, 4,000 hertz or 4,000 cycles per second). The width of the band is called, not surprisingly, the **bandwidth**. FM radio stations also use FDM to share the frequency spectrum between 88 MHz and 108 MHz.

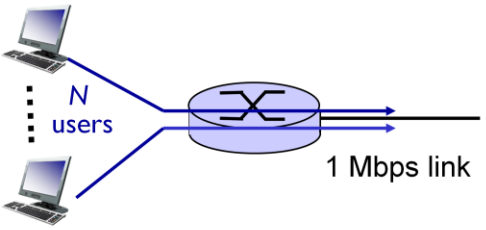
For a TDM link, time is divided into frames of fixed duration, and each frame is divided into a fixed number of time slots. When the network establishes a connection across a link, the network dedicates one time slot in every frame to this connection. These slots are dedicated for the sole use of that connection, with one time slot available for use (in every frame) to transmit the connection's data.

Packet switching versus circuit switching

packet switching allows more users to use network!

example:

- 1 Mb/s link
- each user:
  - 100 kb/s when "active"
  - active 10% of time



- *circuit-switching:*
  - 10 users
- *packet switching:*
  - with 35 users, probability > 10 active at same time is less than .0004 *

Q: how did we get value 0.0004?

Q: what happens if > 35 users ?

* Check out the online interactive exercises for more examples

Introduction 1-30

Proponents of circuit switching say that packet switching wont work for real-time applications because of variable queueing delays. Proponents of packet switching have always argued that circuit switching is wasteful because the dedicated circuits are idle during **silent periods.** Proponents of packet switching argue that (1) it offers better sharing of transmission capacity than circuit switching and (2) it is simpler, more efficient, and less costly to implement than circuit switching.

Why is packet switching more efficient? Let's look at a simple example. EXAMPLE Suppose users share a 1 Mbps link. Also suppose that each user alternates between periods of activity; when a user generates data at a constant rate of 100 kbps, and periods of inactivity, when a user generates no data. Suppose further that a user is active only 10 percent of the time. With circuit switching, 100 kbps must be *reserved* for *each* user at all times. For example, with circuit-switched TDM, if a one-second frame is divided

into 10 time slots of 100 ms each, then each user would be allocated one time slot per frame. Thus, the circuit-switched link can support only 10 (= 1 Mbps/100 kbps) simultaneous users.

With packet switching, the probability that a specific user is active is 0.1 (that is, 10 percent). If there are 35 users, the probability that there are 11 or more simultaneously active users is approximately 0.0004. (see problem 8 end of

chapter 1). When there are 10 or fewer simultaneously active users (which happens with probability 0.9996), the aggregate arrival rate of data is less than or equal to 1 Mbps, the output rate of the link. Thus, when there are 10 or fewer active users, users' packets flow through the link essentially without delay, as is the case with circuit switching. When there are more than 10 simultaneously active users, then the aggregate arrival rate of packets exceeds the output capacity of the link, and the output queue will begin to grow. (It continues to grow until the aggregate input rate falls back below 1 Mbps, at which point the

queue will begin to diminish in length.) Because the probability of having more than 10 simultaneously active users is minuscule in this example, packet switching provides essentially the same performance as circuit switching, *but does so while allowing for more than three times the number of users.*

Packet switching versus circuit switching

is packet switching a "slam dunk winner?"

- great for bursty data
  - resource sharing
  - simpler, no call setup
- excessive congestion possible: packet delay and loss
  - protocols needed for reliable data transfer, congestion control
- *Q:* How to provide circuit-like behavior?
  - bandwidth guarantees needed for audio/video apps
  - still an unsolved problem

Q: human analogies of reserved resources (circuit switching) versus on-demand allocation (packet-switching)?

GREAT FOR BURSTY DATA Let's now consider a second simple example. Suppose there are 10 users and that one user suddenly generates one thousand 1,000-bit packets, while other users remain quiescent and do not generate packets. Under TDM circuit switching with 10 slots per frame and each slot consisting of 1,000 bits, the active user can only use its one time slot per frame to transmit data, while the remaining nine time slots in each frame remain idle. It will be 10 seconds before all of the active user's one million bits of data has been transmitted. In the case of packet switching, the active user can continuously send its packets at the full link rate of 1 Mbps, since there are no other users generating packets that need to be multiplexed with the active user's packets. In this case, all of the active user's data will be transmitted within 1 second.

READ SLIDE

Internet structure: network of networks

- End systems connect to Internet via access ISPs (Internet Service Providers)
  - Residential, company and university ISPs
- Access ISPs in turn must be interconnected.
  - So that any two hosts can send packets to each other
- Resulting network of networks is very complex
  - Evolution was driven by economics and national policies
- Let's take a stepwise approach to describe current Internet structure

We saw earlier that end systems (PCs, smartphones, Web servers, mail servers, and so on) connect into the Internet via an access ISP. The access ISP can provide either wired or wireless connectivity, using an array of access technologies including DSL, cable, FTTH, Wi-Fi, and cellular. Note that the access ISP does not have to be a telcom or a cable company; instead it can be, for example, a university (providing Internet access to students, staff, and faculty), or a company (providing access for its employees).

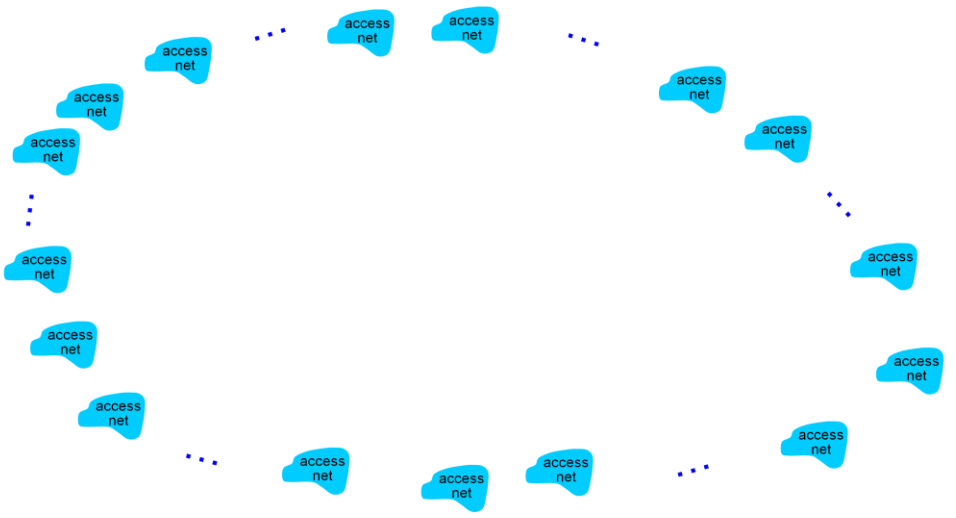
>But connecting end users and content providers into an access ISP is only a small piece of solving the puzzle of connecting the billions of end systems that make up the internet. To complete this puzzle, the access ISPs themselves must be interconnected.

>This is done by creating a *network of networks*—understanding this phrase is the key to understanding the Internet. Over the years, the network of networks that forms the Internet has evolved into a very complex structure. Much of this evolution is driven by economics and national policy, rather than by performance considerations.

In order to understand today's Internet network structure, let's incrementally build a series of network structures, with each new structure being a better approximation of the complex Internet that we have today.

Internet structure: network of networks

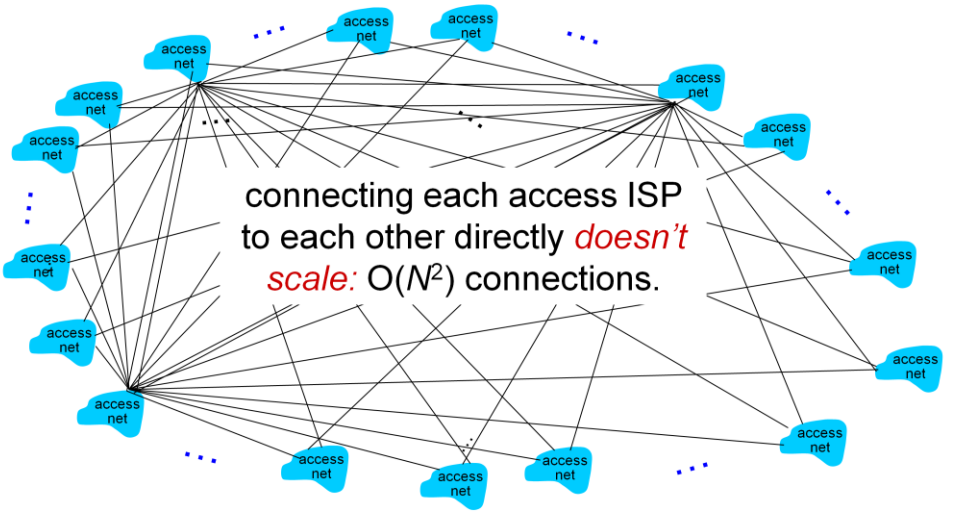
Question: given *millions* of access ISPs, how to connect them together?



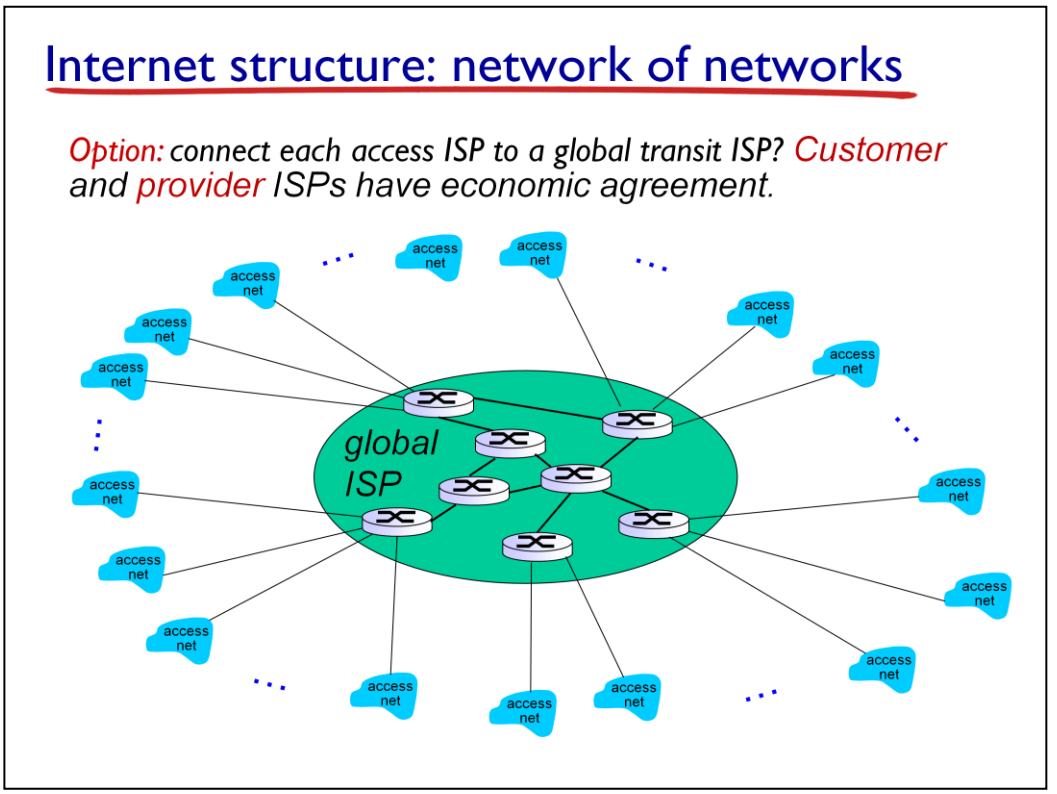
JUST READ SLIDE

Internet structure: network of networks

Option: connect each access ISP to every other access ISP?



Recall that the overarching goal is to interconnect the access ISPs so that all end systems can send packets to each other. One naïve approach would be to have each access ISP *directly* connect with every other access ISP. Such a mesh design is, of course, much too costly for the access ISPs, as it would require each access ISP to have a separate communication link to each of the hundreds of thousands of other access ISPs all over the world.

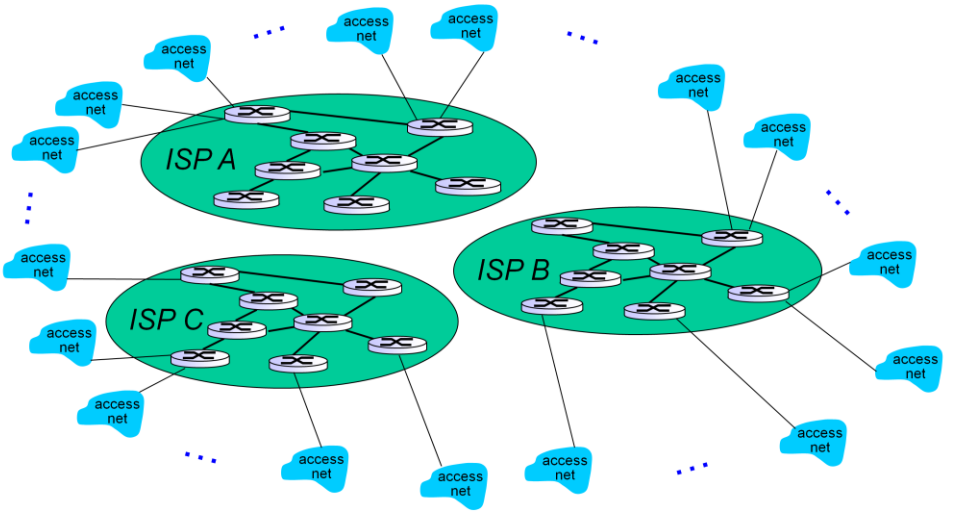


Let's try something else:

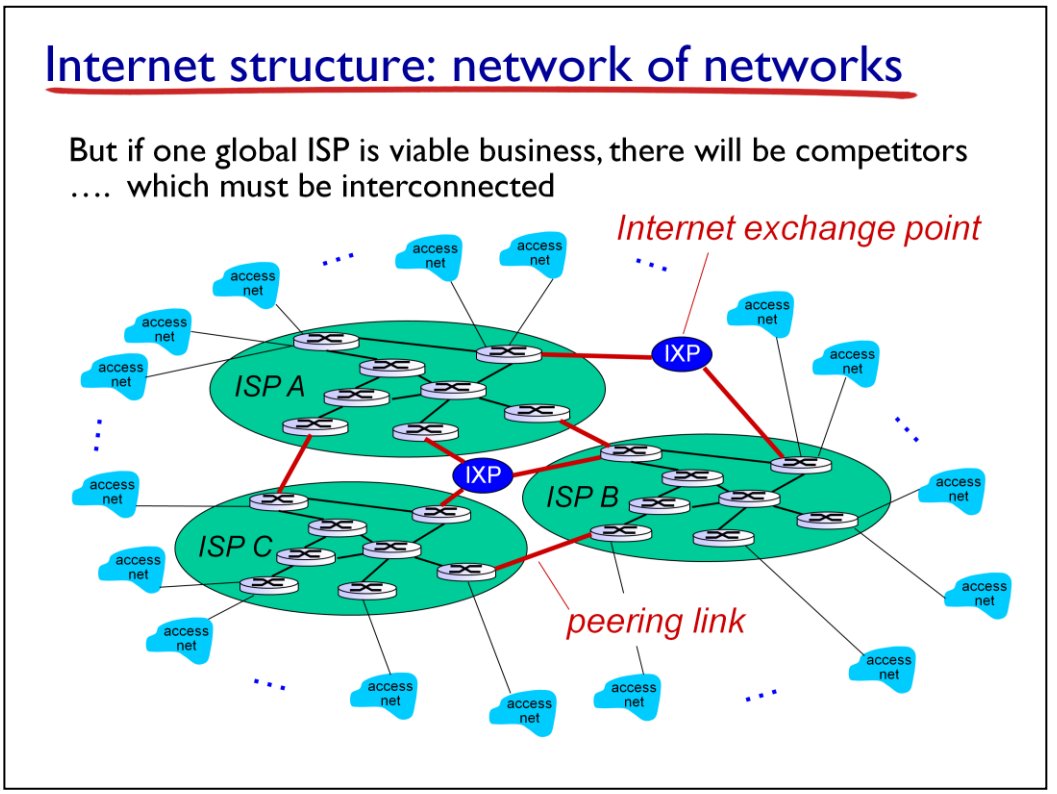
Lets first try a network like this that interconnects all of the access ISPs with a *single global transit ISP – lets call this* ***Network Structure 1***. Our (imaginary) global transit ISP is a network of routers and communication links that not only spans the globe, but also has at least one router near each of the hundreds of thousands of access ISPs. Of course, it would be very costly for the global ISP to build such an extensive network. To be profitable, it would naturally charge each of the access ISPs for connectivity, with the pricing reflecting (but not necessarily directly proportional to) the amount of traffic an access ISP exchanges with the global ISP. Since the access ISP pays the global transit ISP, the access ISP is said to be a **customer** and the global transit ISP is said to be a **provider**.

Internet structure: network of networks

But if one global ISP is viable business, there will be competitors ….



Now if some company builds and operates a global transit ISP that is profitable, then it is natural for other companies to build their own global transit ISPs and compete with the original global transit ISP. This leads to our second network description, which consists of the hundreds of thousands of access ISPs and *multiple* global transit ISPs. The access ISPs certainly prefer this second network design over the first since they can now choose among the competing global transit providers as a function of their pricing and services.

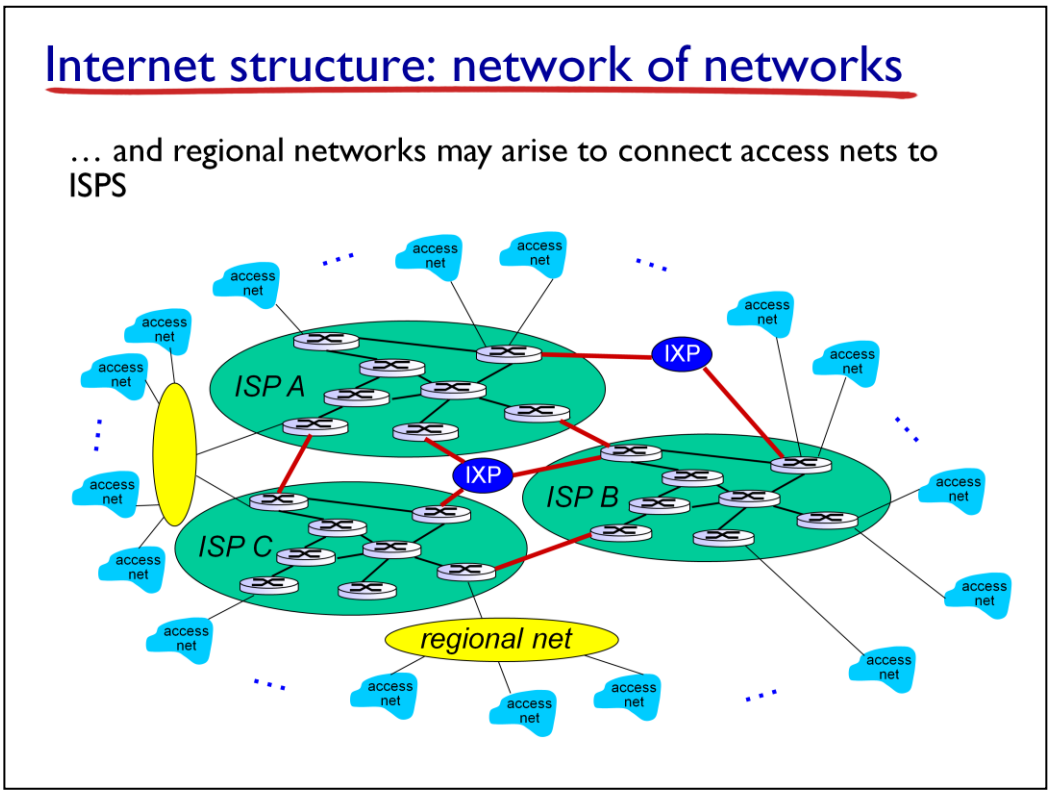


Note, however, that the global transit ISPs. themselves must interconnect: Otherwise access ISPs connected to one of the global transit providers would not be able to communicate with access ISPs connected to the other global transit providers.

This network description is a two-tier hierarchy with global transit providers residing at the top tier and access ISPs at the bottom tier. This assumes that global transit ISPs are not only capable of getting close to each and every access ISP, but also find it economically desirable to do so.

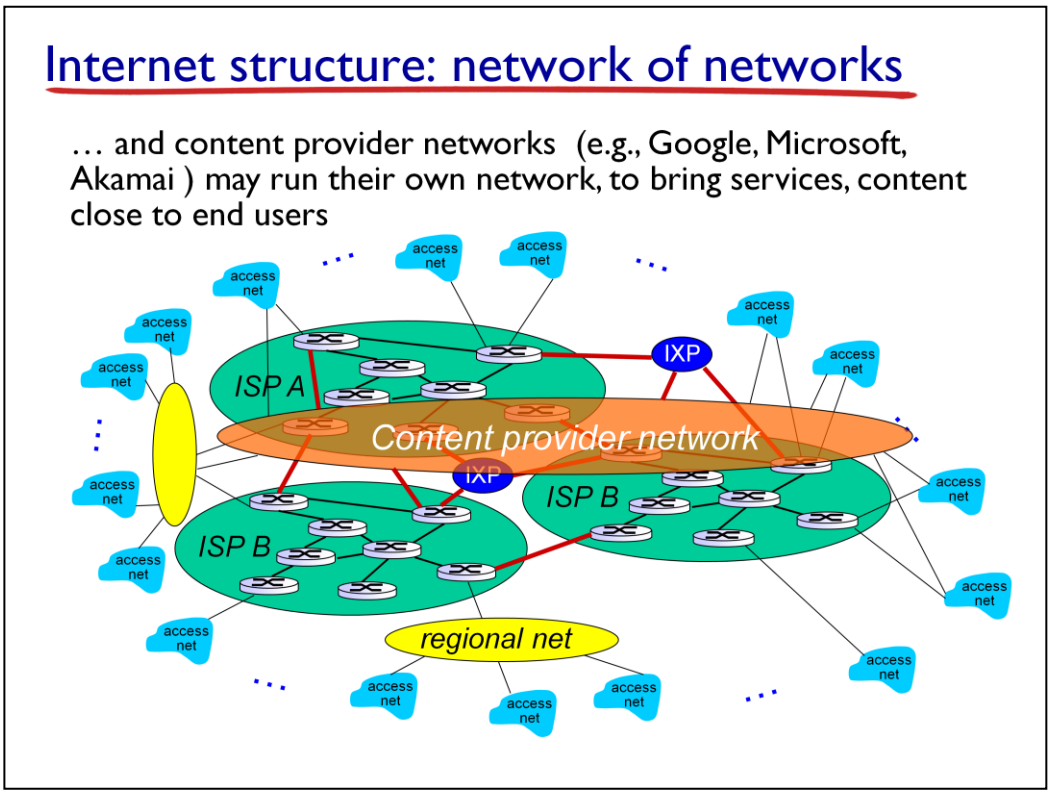
To reduce these costs, a pair of global ISPs can **peer**, that is, they can directly connect their networks together so that all the traffic between them passes over the direct connection rather than through upstream intermediaries. This is typically settlement-free, that is, neither ISP pays the other.

Along these same lines, a third-party company can create an **Internet Exchange Point (IXP)** (typically in a stand-alone building with its own switches), which is a meeting point where multiple ISPs can peer together. There are roughly 300 IXPs in the Internet today [Augustin 2009].



In reality, although some ISPs do have impressive global coverage and do directly connect with many access ISPs, no ISP has presence in each and every city in the world. Instead, in any given region, there may be a **regional ISP** to which the access ISPs in the region connect. Each regional ISP then connects to **tier-1 ISPs**. Tier-1 ISPs are similar to our (imaginary) global transit ISP; but tier-1 ISPs, which actually do exist, do not have a presence in every city in the world. There are approximately a dozen tier-1 ISPs, including Level 3 Communications, AT&T, Vodafone.

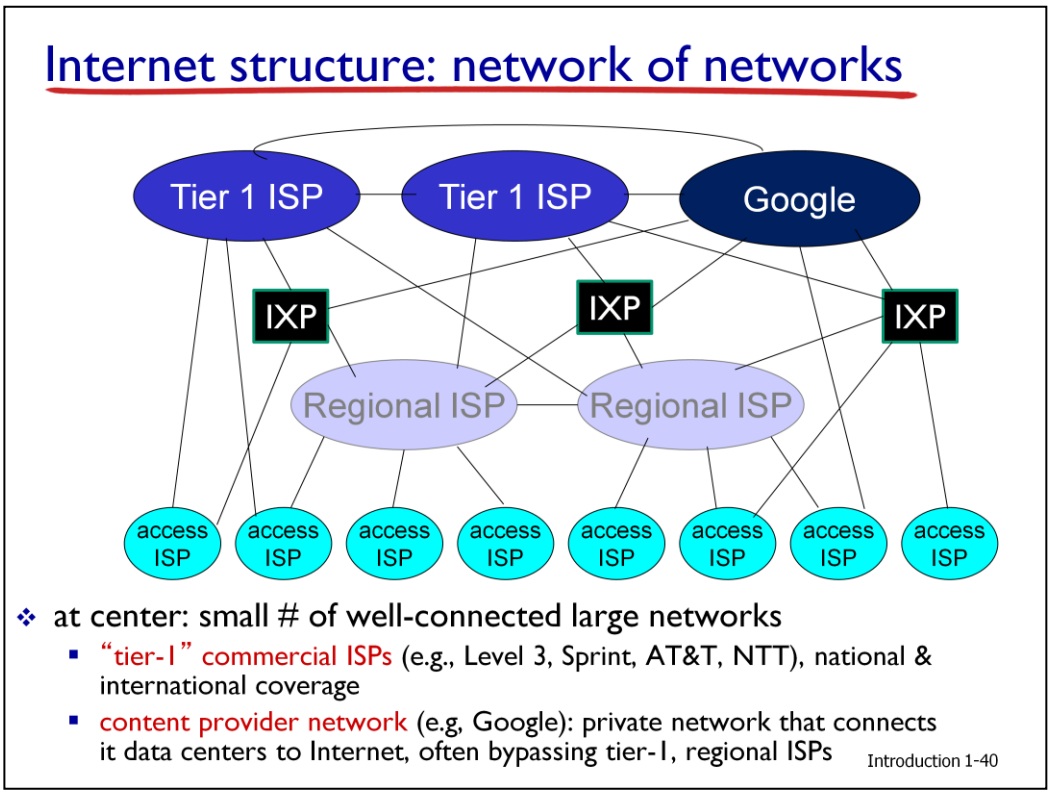
In fact, there may be multiple competing regional ISPs in a region. In such a hierarchy, each access ISP pays the regional ISP to which it connects, and each regional ISP pays the tier-1 ISP to which it connects. (An access ISP can also connect directly to a tier-1 ISP, in which case it pays the tier-1 ISP). Thus, there is customer-provider relationship at each level of the hierarchy. Note that the tier-1 ISPs do not pay anyone as they are at the top of the hierarchy. To further complicate matters, in some regions, there may be a larger regional ISP (possibly spanning an entire country) to which the smaller regional ISPs in that region connect; the larger regional ISP then connects to a tier-1 ISP. For example, in China, there are access ISPs in each city, which connect to provincial ISPs, which in turn connect to national ISPs, which finally connect to tier-1 ISPs. We refer to this multi-tier hierarchy, which is still only a crude approximation of today's Internet, as *Network Structure 3*.



Finally we arrive here. This internet structure describes the Internet of today. It builds on the previous slide by adding **content provider networks**. Google is currently one of the leading examples of such a content provider network. It is estimated that Google has 30 to 50 data centers distributed across North America, Europe, Asia, South America, and Australia.

The Google data centers are all interconnected via Google's private TCP/IP network, which spans the entire globe but is nevertheless separate from the public Internet.

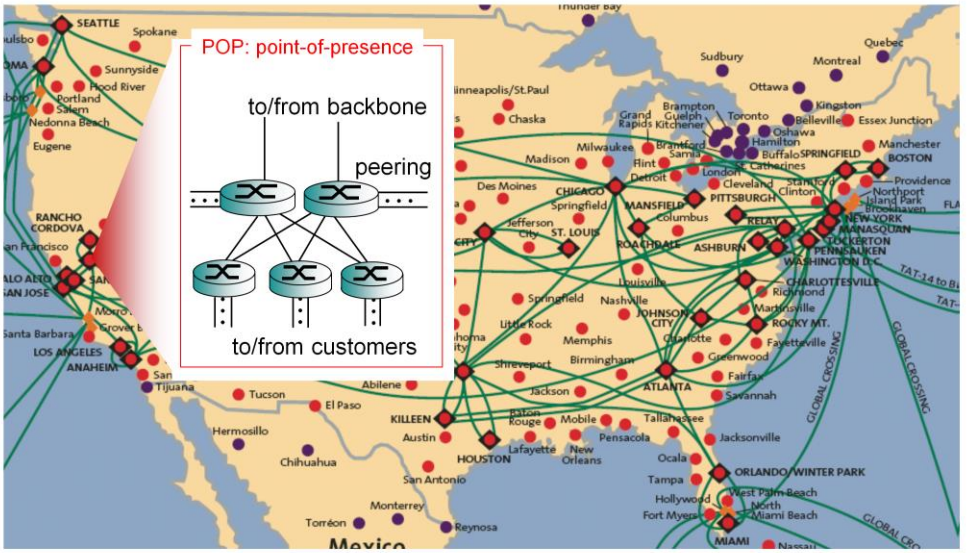
Importantly, the Google private network only carries traffic to/from Google servers. As shown in this slide, the Google private network attempts to "bypass" the upper tiers of the Internet by peering (settlement free) with lower-tier ISPs, either by directly connecting with them or by connecting with them at IXPs. However, because many access ISPs can still only be reached by transiting through tier-1 networks, the Google network also connects to tier-1 ISPs, and pays those ISPs for the traffic it exchanges with them. By creating its own network, a content provider not only reduces its payments to upper-tier ISPs, but also has greater control of how its services are ultimately delivered to end users.



Summary

In summary, today's Internet—a network of networks—is complex, consisting of a dozen or so tier-1 ISPs and hundreds of thousands of lower-tier ISPs. The ISPs are diverse in their coverage, with some spanning multiple continents and oceans, and others limited to narrow geographic regions. The lower-tier ISPs connect to the higher-tier ISPs, and the higher-tier ISPs interconnect with one another. Users and content providers are customers of lower-tier ISPs, and lower-tier ISPs are customers of higher-tier ISPs. In recent years, major content providers have also created their own networks and connect directly into lower-tier ISPs where possible.

Tier-1 ISP: e.g., Sprint



Introduction 1-41

PoPs exist in all levels of the hierarchy, except for the bottom (access ISP) level. A **PoP** is simply a group of one or more routers (at the same location) in the provider's network where customer ISPs can connect into the provider ISP. For a customer network to connect to a provider's PoP, it can lease a high-speed link from a third-party telecommunications provider to directly connect one of its routers to a router at the PoP.

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

1.6 networks under attack: security

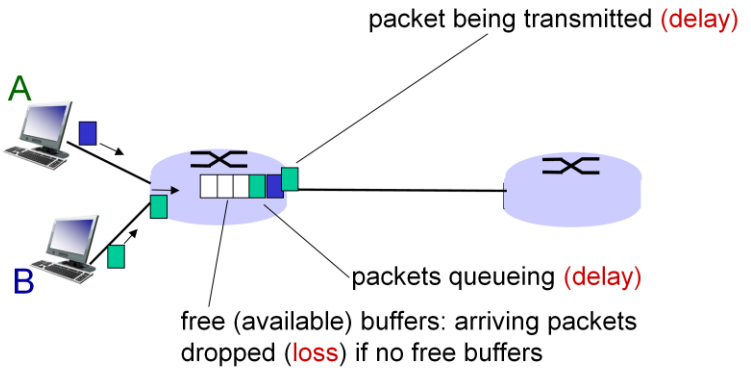
1.7 history

Previously said the internet could be viewed as an infrastructure that provides services to distributed applications running on end systems. Ideally, we would like Internet services to be able to move as much data as we want between any two end systems, instantaneously, without any loss of data. Alas, this is a lofty goal, one that is unachievable in reality. Instead, computer networks necessarily constrain **throughput** (the amount of data per second that can be transferred). between end systems, introduce delays between end systems, and can actually lose packets. Next we'll begin to examine and quantify delay, loss, and throughput in computer networks.

How do loss and delay occur?

packets *queue* in router buffers

- packet arrival rate to link (temporarily) exceeds output link capacity
- packets queue, wait for turn



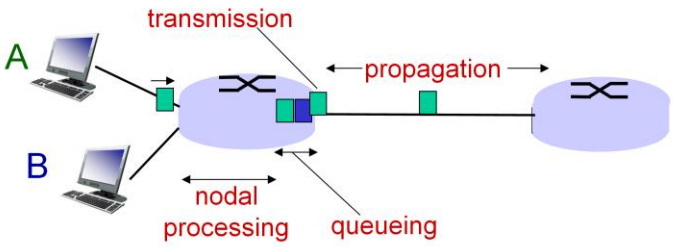
Introduction 1-43

Recall that a packet starts in a host (the source), passes through a series of routers, and ends its journey in another host (the destination). As it makes that jounrey it suffers a number of different types of delay. Let's quickly revisit one type – queuing delay.

SEE FIGURE As part of its end-to-end route between source and destination, a packet is sent from the upstream nodes A and B through the left router to right router. Our goal is to characterize the nodal delay at the first router. When the packet arrives at the left router from the upstream node A or B, te router examines the packet's header to determine the appropriate outbound link for the packet and then directs the packet to this link. In this example, the outbound link for the packet is the one that leads to the router on the right. A packet can be transmitted on a link only if there is no other packet currently being transmitted on the link and if there are no other packets preceding it in the queue; if the link is currently busy or if there are other packets already queued for the link, the newly arriving packet will then join the queue.

This is just one type of delay..there are more that we must consider in order to fully characterise the total nodal delay (i.e. the total delay that happens at the first router)

Four sources of packet delay



$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

d_{proc}: nodal processing

- check bit errors
- determine output link
- typically < msec

d_{queue}: queueing delay

- time waiting at output link for transmission
- depends on congestion level of router

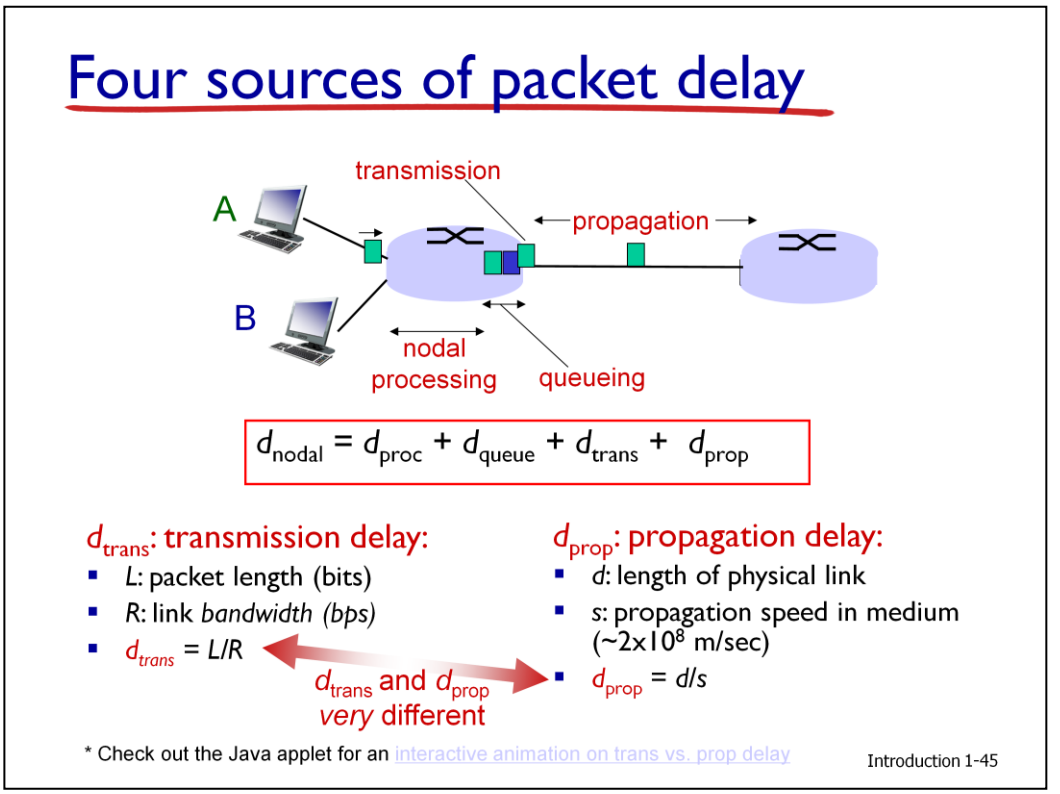
Introduction 1-44

Recall that a packet starts in a host (the source), passes through a series of routers, and ends its journey in another host (the destination). As a packet travels from one node (host or router) to the subsequent node (host or router) along this path, the packet suffers from several types of delays at *each* node along the path. The most important of these delays are the **nodal processing delay, queuing delay, transmission delay,** and **propagation delay**; together, these delays accumulate to give a **total nodal delay**..lets start by looking at queuing delay.

The time required to examine the packet's header and determine where to direct the packet is part of the **processing delay**. The processing delay can also include other factors, such as the time needed to check for bit-level errors in the packet that occurred in transmitting the packet's bits from the upstream node to the router. Processing delays in high-speed routers are typically on the order of microseconds or less. After this nodal processing, the router directs the packet to the queue that precedes the link to the next router.

At the queue, the packet experiences a **queuing delay** as it waits to be transmitted onto the link. The length of the queuing delay of a specific packet will depend on the number of earlier-arriving packets that are queued and waiting for transmission onto the link. If the queue is empty and no other packet is currently being transmitted, then our packet's queuing delay will be

zero. On the other hand, if the traffic is heavy and many other packets are also waiting to be transmitted, the queuing delay will be long. We will see shortly that the number of packets that an arriving packet might expect to find is a function of the intensity and nature of the traffic arriving at the queue. Queuing delays can be on the order of microseconds to milliseconds in practice.



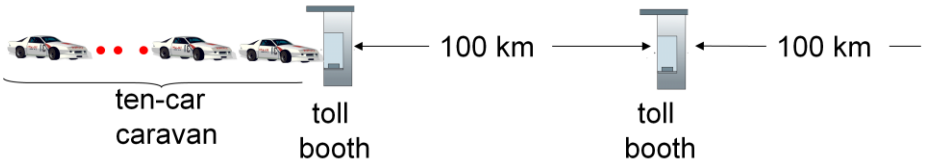
We've moved to the front of the queue….the link is available.

Denote the length of the packet by L bits, and denote the transmission rate of the link from one router to the next by R bits/sec. For example, for a 10 Mbps Ethernet link, the rate is R = 10 Mbps; for a 100 Mbps Ethernet link, the rate is R = 100 Mbps. The **transmission delay** is L/R. This is the amount of time required to push (that is, transmit) all of the packet's bits into the link. Transmission delays are typically on the order of microseconds to milliseconds in practice.

Once a bit is pushed into the link, it needs to propagate to the seocnd router. The time required to propagate from the beginning of the link to this router is the **propagation delay**. The bit propagates at the propagation speed of the link. The propagation speed depends on the physical medium of the link (that is, fiber optics, twisted-pair copper wire, and so on) and is in the range of 2x10^8 meters/sec to 3x10^8 meters/sec which is equal to, or a little less than, the speed of light. The **propagation delay** is the distance between two routers divided by the propagation speed. That is, the propagation delay is d/s, where d is the distance between the routers and s is the propagation speed of the link.

Once the last bit has been received by the second router the process repeats.

Caravan analogy

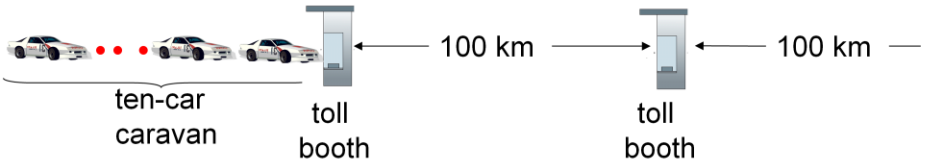


- cars "propagate" at 100 km/hr
- toll booth takes 12 sec to service car (bit transmission time)
- car~bit; caravan ~ packet
- *Q:* How long until caravan is lined up before 2nd toll booth?

- time to "push" entire caravan through toll booth onto highway = 12*10 = 120 sec
- time for last car to propagate from 1st to 2nd toll both: 100km/(100km/hr)= 1 hr
- *A:* 62 minutes

Newcomers to the field of computer networking sometimes have difficulty understanding the difference between transmission delay and propagation delay. The difference is subtle but important. The transmission delay is the amount of time required for the router to push out the packet; it is a function of the packet's length and the transmission rate of the link, but has nothing to do with the distance between the two routers. The propagation delay, on the other hand, is the time it takes a bit to propagate from one router to the next; it is a function of the distance between the two routers, but has nothing to do with the packet's length or the transmission rate of the link.

Caravan analogy (more)

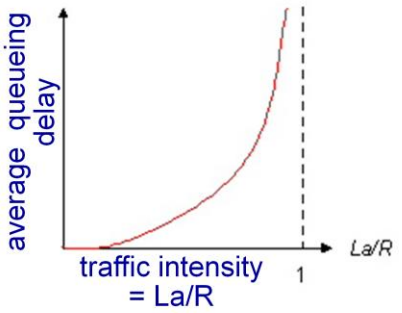


- suppose cars now "propagate" at 1000 km/hr
- and suppose toll booth now takes one min to service a car
- *Q:* Will cars arrive to 2nd booth before all cars serviced at first booth?
  - *A: Yes!* after 7 min, 1st car arrives at second booth; three cars still at 1st booth.

This situation also arises in packet-switched networks—the first bits in a packet can arrive at a router while many of the remaining bits in the packet are still waiting to be transmitted by the preceding router.

Queueing delay (revisited)

- *R:* link bandwidth (bps)
- *L:* packet length (bits)
- a: average packet arrival rate



- *La/R* ~ 0: avg. queueing delay small
- *La/R* -> 1: avg. queueing delay large
- *La/R* > 1: more "work" arriving than can be serviced, average delay infinite!

* Check out the Java applet for an interactive animation on queuing and loss

Introduction 1-48

Unlike the other delays, queing delay can vary from packet to packet. MANY PACKETS ARRIVE AT THE SAME TIME…. So, when is the queuing delay large and when is it insignificant? The answer to this question depends on the rate at which traffic arrives at the queue, the transmission rate of the link, and the nature of the arriving traffic, that is, whether the traffic arrives periodically or arrives in bursts.

To gain some insight here, let *a* denote the average rate at which packets arrive at the queue (*a* is in units of packets/sec). Recall that *R* is the transmission rate; that is, it is the rate (in bits/sec) at which bits are pushed out of the queue. Also suppose, for simplicity, that all packets consist of *L* bits. Then the average rate at which bits arrive at the queue is *La* bits/sec.

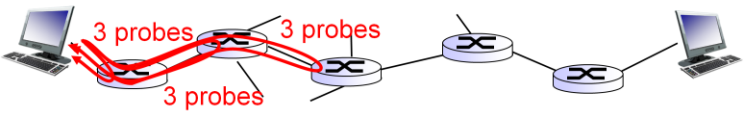
Finally, assume that the queue is very big, so that it can hold essentially an infinite number of bits. The ratio *La/R*, called the **traffic intensity**, often plays an important role in estimating the extent of the queuing delay. If *La/R* > 1, then the average rate at which bits arrive at the queue exceeds the rate at which the bits can be transmitted from the queue. In this unfortunate situation, the queue will tend to increase without bound and the queuing delay will approach infinity! Therefore, one of the golden rules in traffic engineering is: *Design your system so that the traffic intensity is no greater than 1.*

Now consider the case $La/R \leq 1$. Here, the nature of the arriving traffic impacts the queuing delay. For example, if packets arrive periodically—that is, one packet arrives every L/R seconds—then every packet will arrive at an empty queue and there will be no queuing delay. On the other hand, if packets arrive in bursts but periodically, there can be a significant average queuing delay. For example, suppose N packets arrive simultaneously every $(L/R)N$ seconds. Then the first packet transmitted has no queuing delay; the second packet transmitted has a queuing delay of L/R seconds; and more generally, the nth packet transmitted has a queuing delay of $(n-1)L/R$ seconds. We leave it as an exercise for you to calculate the average queuing delay in this example.

Typically, the arrival process to a queue is *random;* Nonetheless, as the traffic intensity approaches 1, the average queue length gets larger and larger. The qualitative dependence of average queuing delay on the traffic intensity is shown in the figure.

"Real" Internet delays and routes

- what do "real" Internet delay & loss look like?
- `traceroute` program: provides delay measurement from source to router along end-end Internet path towards destination. For all *i:*
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply.

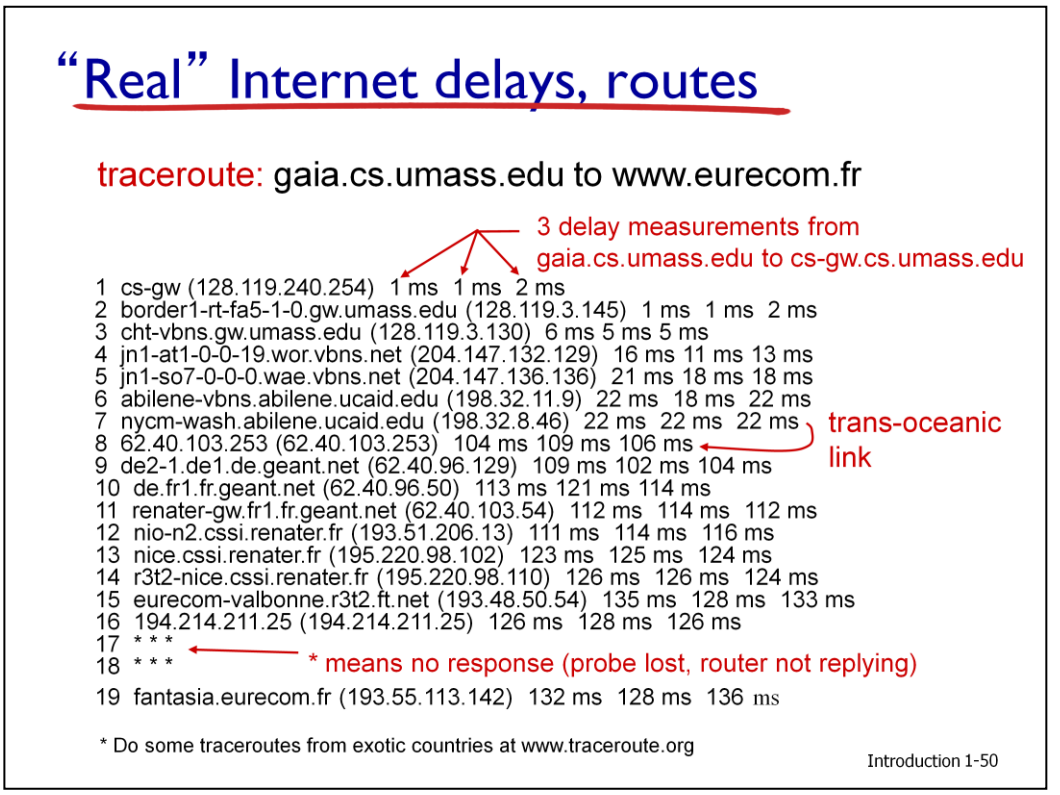


Introduction 1-49

To get a hands-on feel for end-to-end delay in a computer network, we can make use of the Traceroute program. Traceroute can run in any Internet host. When the user specifies a destination hostname, the program in the source host sends multiple, special packets toward that destination. As these packets work their way toward the destination, they pass through a series of routers. When a router receives one of these special packets, it sends back to the source a short message that contains the name and address of the router.

More specifically, suppose there are N -1 routers between source and destination. Then the source will send N special packets into the network, with each packet addressed to the ultimate destination. These N special packets are marked *1* through N, with the first packet marked *1* and the last packet marked N. When the nth router receives the nth packet marked n, the router does not forward the packet toward its destination, but instead sends a message back to the source. When the destination host receives the Nth packet, it too returns a message back to the source. The source records the time that elapses between when it sends a packet and when it receives the corresponding return message; it also records the name and address of the router (or the destination host) that returns the message.

Traceroute actually repeats the experiment just described three times, so the source actually sends $3 \bullet N$ packets to the destination.

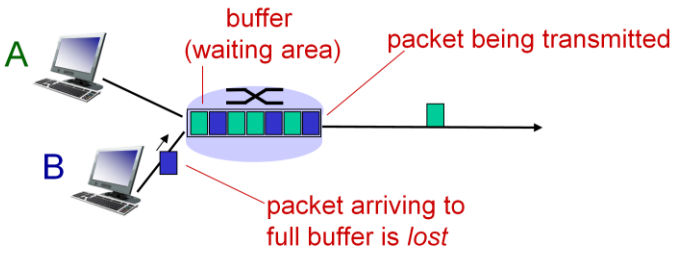


Here is an example of the output of the Traceroute program, where the route was being traced from the source host gaia.cs.umass.edu (at the University of Massachusetts) to the host www.eurecom.fr. The output has six columns: the first column is the *n* value described above, that is, the number of the router along the route; the second column is the name of the router; the third column is the address of the router (of the form xxx.xxx.xxx.xxx); the last three columns are the round-trip delays for three experiments. If the source receives fewer than three messages from any given router (due to packet loss in the network), Traceroute places an asterisk just after the router number and reports fewer than three round-trip times for that router.

Because the queuing delay is varying with time, the round-trip delay of packet *n* sent to a router *n* can sometimes be longer than the round-trip delay of packet $n+1$ sent to router $n+1$.

Packet loss

- queue (aka buffer) preceding link in buffer has finite capacity
- packet arriving to full queue dropped (aka lost)
- lost packet may be retransmitted by previous node, by source end system, or not at all



* Check out the Java applet for an interactive animation on queuing and loss

Introduction 1-51

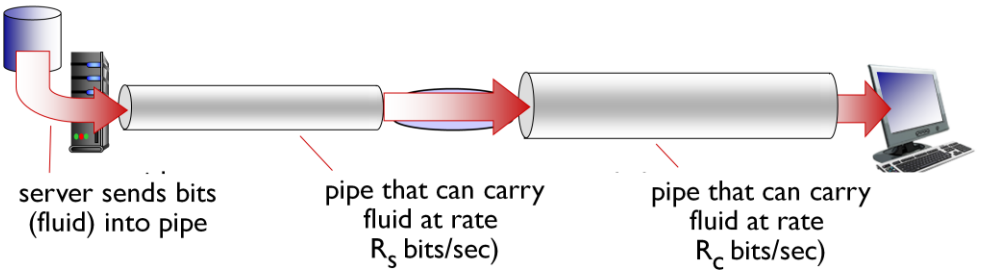
In our discussions above, we have assumed that the queue is capable of holding an infinite number of packets. In reality a queue preceding a link has finite capacity, although the queuing capacity greatly depends on the router design and cost.

Because the queue capacity is finite, packet delays do not really approach infinity as the traffic intensity approaches 1. Instead, a packet can arrive to find a full queue. With no place to store such a packet, a router will **drop** that packet; that is, the packet will be **lost**.

From an end-system viewpoint, a packet loss will look like a packet having been transmitted into the network core but never emerging from the network at the destination. The fraction of lost packets increases as the traffic intensity increases. Therefore, performance at a node is often measured not only in terms of delay, but also in terms of the probability of packet loss.

Throughput

- *throughput:* rate (bits/time unit) at which bits transferred between sender/receiver
  - *instantaneous:* rate at given point in time
  - *average:* rate over longer period of time



Introduction 1-52

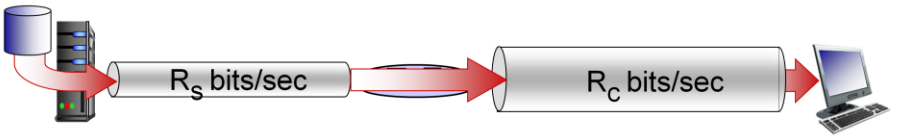
In addition to delay and packet loss, another critical performance measure in computer networks is end-to-end throughput. To define throughput, consider transferring a large file from Host A to Host B across a computer network. This transfer might be, for example, a large video clip from one peer to another in a P2P file sharing system.

The **instantaneous throughput** at any instant of time is the rate (in bits/sec) at which Host B is receiving the file. (Many applications, including many P2P file sharing systems, display the instantaneous throughput during downloads in the user interface—perhaps you have observed this before!) If the file consists of F bits and the transfer takes T seconds for Host B to receive all F bits, then the **average throughput** of the file transfer is F/T bits/sec.

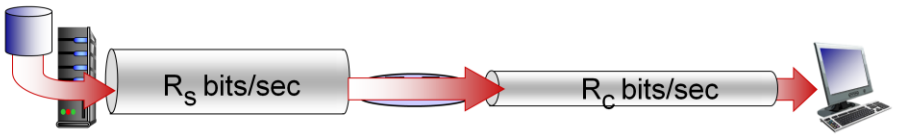
For some applications, such as Internet telephony, it is desirable to have a low delay and an instantaneous throughput consistently above some threshold (for example, over 24 kbps for some Internet telephony applications and over 256 kbps for some realtime video applications). For other applications, including those involving file transfers, delay is not critical, but it is desirable to have the highest possible throughput.

Throughput (more)

- $R_s < R_c$ What is average end-end throughput?



- $R_s > R_c$ What is average end-end throughput?



bottleneck link

link on end-end path that constrains end-end throughput

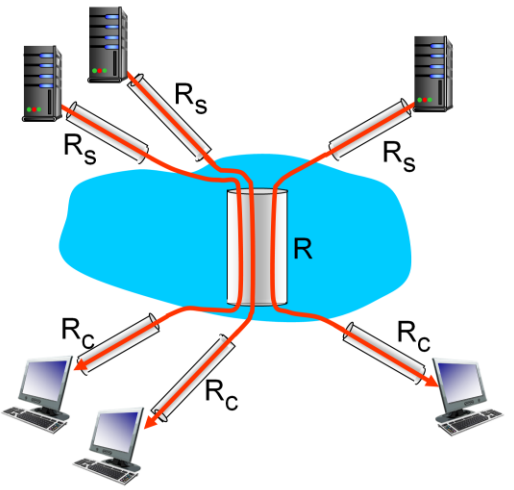
Introduction 1-53

To gain further insight into the important concept of throughput, let's consider a few examples. The figure shows two end systems, a server and a client, connected by two communication links and a router. Consider the throughput for a file transfer from the server to the client. Let *Rs* denote the rate of the link between the server and the router; and *Rc* denote the rate of the link between the router and the client.

>Suppose that the only bits being sent in the entire network are those from the server to the client. We now ask, in this ideal scenario, what is the serverto-client throughput? To answer this question, we may think of bits as *fluid* and communication links as *pipes*. Clearly, the server cannot pump bits through its link at a rate faster than *Rs* bps; and the router cannot forward bits at a rate faster than *Rc* bps. If *Rs* < *Rc*, then the bits pumped by the server will "flow" right through the router and arrive at the client at a rate of *Rs* bps, giving a throughput of *Rs* bps. If, on the other hand, *Rc* < *Rs*, then the router will not be able to forward bits as quickly as it receives them. In this case, bits will only leave the router at rate *Rc*, giving an end-to-end throughput of *Rc*. (Note also that if bits continue to arrive at the router at rate *Rs*, and continue to leave the router at *Rc*, the backlog of bits at the router waiting for transmission to the client will grow and grow—a most undesirable situation!) Thus, for this simple two-link network, the throughput is min{*Rc*, *Rs*}, that is, it is the transmission rate of the **bottleneck link**. Having determined the throughput, we can now approximate the time it takes to transfer a large file of *F* bits from server to client as *F*/min{*Rs*, *Rc*}.

Throughput: Internet scenario

- per-connection end-end throughput: $\min(R_c, R_s, R/10)$
- in practice: R_c or R_s is often bottleneck



10 connections (fairly) share backbone bottleneck link R bits/sec

Introduction 1-54

For a final example, consider the case in which there are 10 servers and 10 clients connected to the core of the computer network. In this example, there are 10 simultaneous downloads taking place, involving 10 client-server pairs. Suppose that these 10 downloads are the only traffic in the network at the current time. As shown in the figure, there is a link in the core that is traversed by all 10 downloads. Denote *R* for the transmission rate of this link *R*. Let's suppose that all server access links have the same rate *Rs*, all client access links have the same rate *Rc*, and the transmission rates of all the links in the core—except the one common link of rate *R*—are much larger than *Rs*, *Rc*, and *R*.

Now we ask, what are the throughputs of the downloads? Clearly, if the rate of the common link, *R*, is large—say a hundred times larger than both *Rs* and *Rc*—then the throughput for each download will once again be min{*Rs*, *Rc*}. But what if the rate of the common link is of the same order as *Rs* and *Rc*? What will the throughput be in this case? Let's take a look at a specific example. Suppose *Rs* = 2 Mbps, *Rc* = 1 Mbps, *R* = 5 Mbps, and the common link divides its transmission rate equally among the 10 downloads. Then the bottleneck for each download is no longer in the access network, but is now instead the shared link in the core, which only provides each download with 500 kbps of throughput. Thus the end-to-end throughput for each download is now reduced to 500 kbps. TYPICALLY NOT THE CASE

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

1.6 networks under attack: security

1.7 history

Protocol "layers"

Networks are complex, with many "pieces":

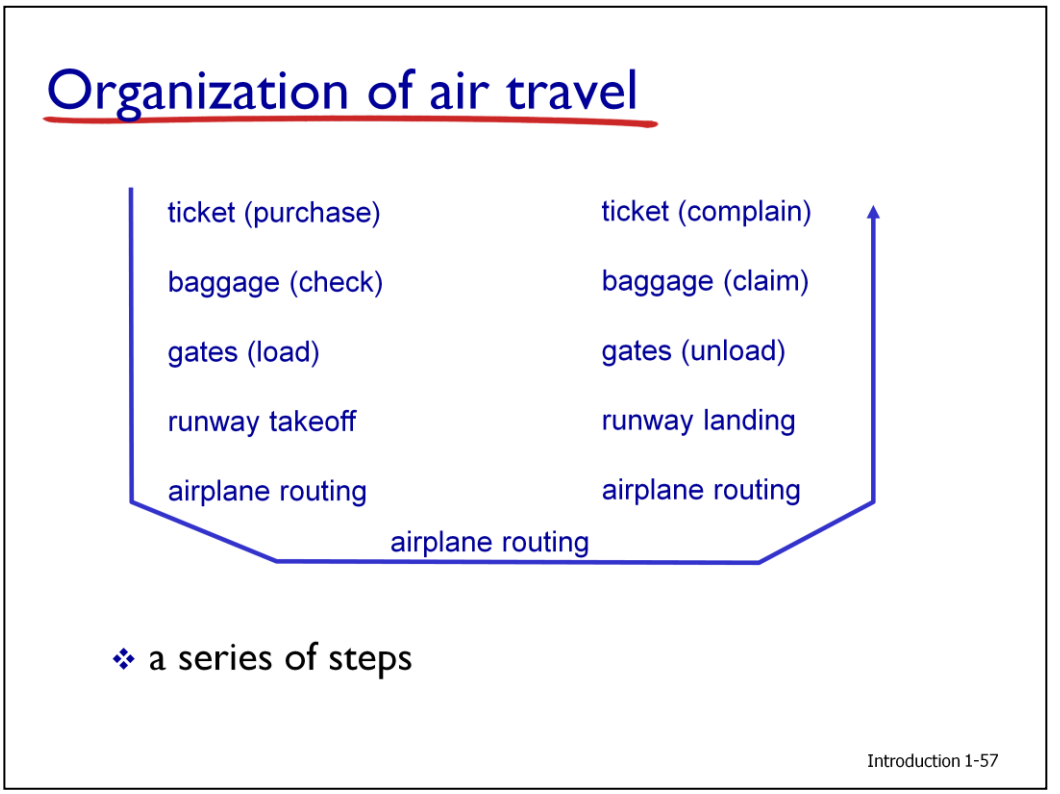
- hosts
- routers
- links of various media
- applications
- protocols
- hardware, software

Question:

is there any hope of *organizing* structure of network?

…. or at least our discussion of networks?

From our discussion thus far, it is apparent that the Internet is an *extremely* complicated system. We have seen that there are many pieces to the Internet: numerous applications and protocols, various types of end systems, packet switches, and various types of link-level media. Given this enormous complexity, is there any hope of organizing a network architecture, or at least our discussion of network architecture? Fortunately, the answer to both questions is yes. Network engineers have come up with a very useful way of thinking of the intenet of being broken down into a set of independent layers, each with its own set of rules or protocols.

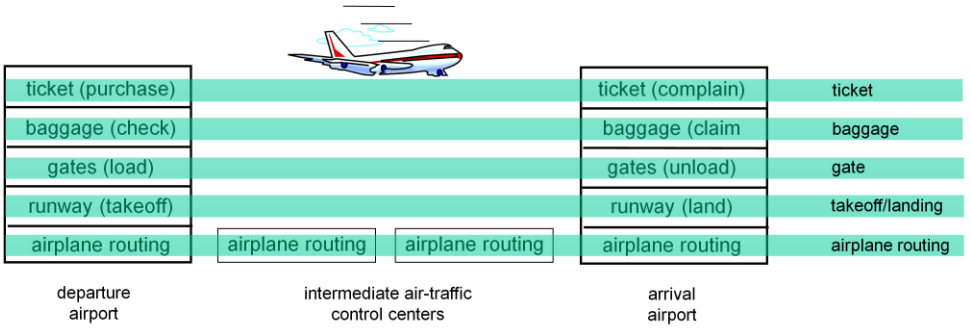


Before attempting to organize our thoughts on Internet architecture, let's look for a human analogy. Imagine if someone asked you to describe, for example, the airline system. How would you find the structure to describe this complex system that has ticketing agents, baggage checkers, gate personnel, pilots, airplanes, air traffic control, and a worldwide system for routing airplanes? One way to describe this system might be to describe the series of actions you take (or others take for you) when you fly on an airline. You purchase your ticket, check your bags, go to the gate, and eventually get loaded onto the plane. The plane takes off and is routed to its destination. After your plane lands, you deplane at the gate and claim your bags. If the trip was bad, you complain about the flight to the ticket agent and so on…

Lets try to put some *structure* on all of this: we note that there is a ticketing function at each end; there is also a baggage function for already-ticketed passengers, and a gate function for already-ticketed and already-baggage-checked passengers. For passengers who have made it through the gate (that is, passengers who are already ticketed, baggage-checked, and through the gate), there is a take-off and landing function, and while in flight, there is an airplane routing function.

>This suggests that we can look This suggests that we can look at the functionality in a *horizontal* manner, as shown in the slide.

Layering of airline functionality



layers: each layer implements a service

- via its own internal-layer actions
- relying on services provided by layer below

Introduction 1-58

In this slide we divide the airline functionality into layers, providing a framework in which we can discuss airline travel. Note that each layer, combined with the layers below it, implements some functionality, some *service.*

>At the ticketing layer and below, airline-counter-to-airline-counter transfer of a person is accomplished.

>At the baggage layer and below, baggage-check-to-baggage-claim transfer of a person and bags is accomplished. Note that the baggage layer provides this service only to an already-ticketed person.

>At the gate layer, departure-gate-to-arrival-gate transfer of a person and bags is accomplished.

>At the takeoff/landing layer, runway-to runway transfer of people and their bags is accomplished.

Each layer provides its service by (1) performing certain actions within that layer (for example, at the gate layer, loading and unloading people from an airplane) and by (2) using the services of the layer directly below it (for example, in the gate layer, using the runway-to runway passenger transfer service of the takeoff/landing layer).

Why layering?

dealing with complex systems:

- explicit structure allows identification, relationship of complex system's pieces
  - layered *reference model* for discussion
- modularization eases maintenance, updating of system
  - change of implementation of layer's service transparent to rest of system
  - e.g., change in gate procedure doesn't affect rest of system
- layering considered harmful?

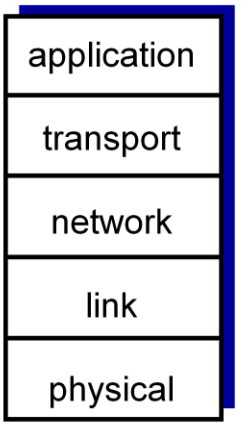
A layered architecture allows us to discuss a well-defined, specific part of a large and complex system. This simplification itself is of considerable value by providing modularity, making it much easier to change the implementation of the service provided by the layer.

As long as the layer provides the same service to the layer above it, and uses the same services from the layer below it, the remainder of the system remains unchanged when a layer's implementation is changed. (Note that changing the implementation of a service is very different from changing the service itself!) For example, if the gate functions were changed (for instance, to have people board and disembark by height), the remainder of the airline system would remain unchanged since the gate layer still provides the same function (loading and unloading people); it simply implements that function in a different manner after the change.

For large and complex systems that are constantly being updated, the ability to change the implementation of a service without affecting other components of the system is another important advantage of layering.

Internet protocol stack

- *application:* supporting network applications - **message**
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer - **segment**
  - TCP, UDP
- *network:* routing of datagrams from source to destination - **datagrams**
  - IP, routing protocols
- *link:* data transfer between neighboring network elements - **frames**
  - Ethernet, 802.111 (WiFi), PPP
- *physical:* **bits** "on the wire"



Introduction 1-60

Application Layer

The application layer is where network applications and their application-layer protocols reside.

The Internet's application layer includes many protocols, such as

> the HTTP protocol (which provides for Web document request and transfer),
>
> SMTP (which provides for the transfer of e-mail messages),
>
> and FTP (which provides for the transfer of files between two end systems).

An application-layer protocol is distributed over multiple end systems, with the application in one end system using the protocol to exchange packets of information with the application in another end system.

We'll refer to this packet of information at the application layer as a **message**.

Transport Layer

The Internet's transport layer transports application-layer messages between application endpoints. In the Internet there are two transport protocols, TCP and UDP, either of which can transport application-layer messages. TCP provides a connection-oriented service to its applications (guaranteed delivery of application-layer messages, flow control (that is, sender/receiver speed matching), breaks into segments. The UDP protocol provides a connectionless no frills service to its applications-no reliability, no flow control, no

congestion control.

Transport-layer packets are commonly referred to as a **segment**.

Network Layer

The Internet's network layer is responsible for moving network-layer packets known as **datagrams** from one host to another.

The Internet transport-layer protocol (TCP or UDP) in a source host passes a transport-layer segment and a destination address to the network layer, just as you would give the postal service a letter with a destination address. The network layer then provides the service of delivering the segment to the transport layer in the destination host.

The Internet's network layer includes the celebrated IP Protocol, which defines the fields in the datagram as well as how the end systems and routers act on these fields. (routing protoclos)

Link Layer

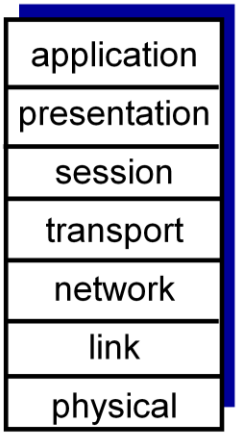
The Internet's network layer routes a datagram through a series of routers between the source and destination. To move a packet from one node (host or router) to the next node in the route, the network layer relies on the services of the link layer. In particular, at each node, the network layer passes the datagram down to the link layer, which delivers the datagram to the next node along the route. At this next node, the link layer passes the datagram up to the network layer. PROTOCOLS Examples of linklayer protocols include Ethernet, WiFi As datagrams typically need to traverse several links to travel from source to destination, a datagram may be handled by different link-layer protocols at different links along its route. The linklayer packets as commonly called **frames**.

Physical Layer

While the job of the link layer is to move entire frames from one network element to an adjacent network element, the job of the physical layer is to move the *individual bits* within the frame from one node to the next. The protocols in this layer are again link dependent and further depend on the actual transmission medium of the link (for example, twisted-pair copper wire, single-mode fiber optics).

ISO/OSI reference model

- *presentation:* allow applications to interpret meaning of data, e.g., encryption, compression, machine-specific conventions
- *session:* synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  - these services, *if needed,* must be implemented in application
  - needed?



In the late 1970s, the International Organization for Standardization (ISO) proposed that computer networks be organized around seven layers, called the Open Systems nterconnection (OSI) model.

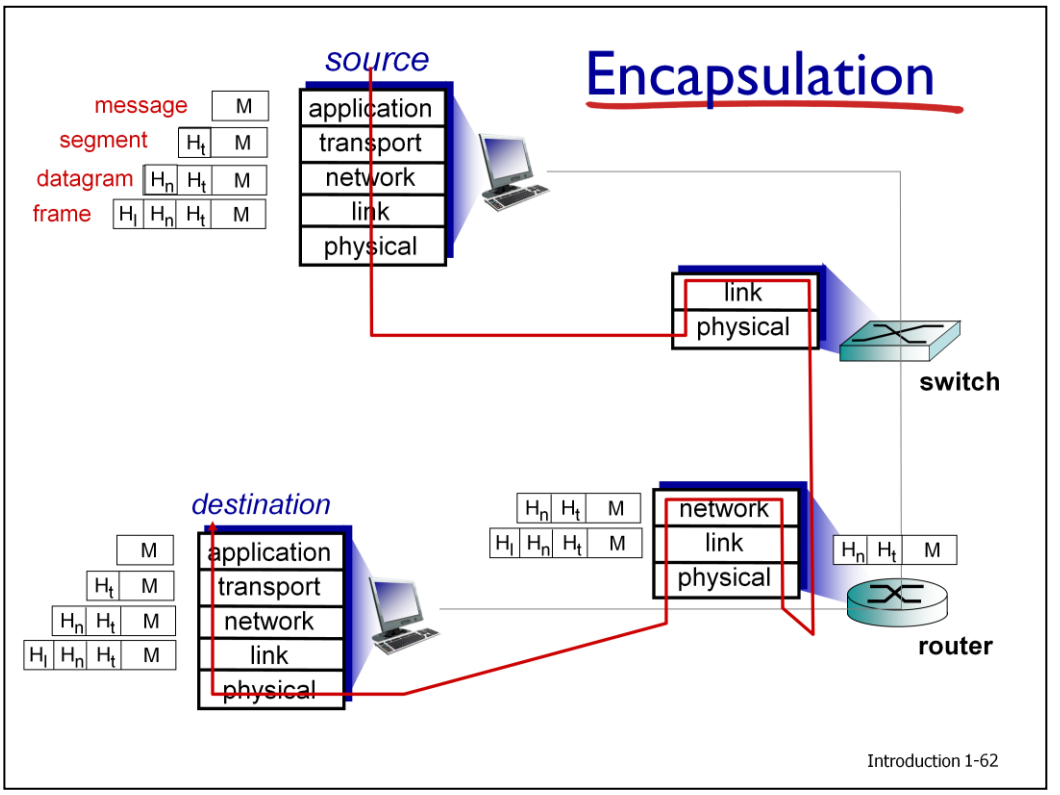
>The OSI model took shape when the protocols that were to become the Internet protocols were in their infancy, and were only one of many different protocol suites under development; in fact, the inventors of the original OSI model probably did not have the Internet in mind when creating it.

>Nevertheless, beginning in the late 1970s, many training and university courses picked up on the ISO mandate and organized courses around the seven-layer model. Because of its early impact on networking education, the seven-layer model continues to linger on in some networking textbooks and training courses.

>Notice the two additional layers present in the OSI reference model—the presentation layer and the session layer.

>The role of the presentation layer is to provide services that allow communicating applications to interpret the meaning of data exchanged. These services include data compression and data encryption.

The session layer provides for delimiting and synchronization of data exchange, including the means to build a checkpointing and recovery scheme.



This slide shows the physical path that data takes down a sending end system's protocol stack, up and down the protocol stacks of an intervening link-layer switch and router, and then up the protocol stack at the receiving end system.

Routers and link-layer switches are both packet switches. Similar to end systems, routers and link-layer switches organize their networking hardware and software into layers.

But routers and link-layer switches do not implement *all* of the layers in the protocol stack; they typically implement only the bottom layers. link-layer switches implement layers 1 and 2; routers implement layers 1 through 3. This means, for example, that Internet routers are capable of implementing the IP protocol (a layer 3 protocol), while link-layer switches are not. We'll see later that while link-layer switches do not recognize IP addresses, they are capable of recognizing layer 2 addresses, such as Ethernet addresses.

Note that hosts implement all five layers; this is consistent with the view that the Internet architecture puts much of its complexity at the edges of the network.

This slide also illustrates the important concept of **encapsulation**.

At the sending host, an **application-layer message** (M) is passed to the transport layer.

START ANIMATIOON

In the simplest case, the transport layer takes the message and appends additional information (so-called transport-layer header information, H*t*) that will be used by the receiver-side transport layer. The application layer message and the transport-layer header information together constitute the **transport-layer segment**.

The transport-layer segment thus encapsulates the application-layer message. The added information might include information allowing the receiver side transport layer to deliver the message up to the appropriate application, and error-detection bits that allow the receiver to determine whether bits in the message have been changed in route.

The transport layer then passes the segment to the network layer, which adds network-layer header information (H*n*) such as source and destination end system addresses, creating a **network-layer datagram**.

The datagram is then passed to the link layer, which (of course!) will add its own link-layer header information and create a **link-layer frame**.

Thus, we see that at each layer, a packet has two types of fields: header fields and a **payload field**. The payload is typically a packet from the layer above.

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

1.6 networks under attack: security

1.7 history

Network security

- field of network security:
  - how bad guys can attack computer networks
  - how we can defend networks against attacks
  - how to design architectures that are immune to attacks
- Internet not originally designed with (much) security in mind
  - *original vision:* "a group of mutually trusting users attached to a transparent network" ☺
  - Internet protocol designers playing "catch-up"
  - security considerations in all layers!

The field of network security is about how hackers (aka the bad guys) can attack computer networks and about how we can defend networks against those attacks, or better yet, design new architectures that are immune to such attacks in the first place.

Given the frequency and variety of existing attacks as well as the threat of new and more destructive future attacks, network security has become a central topic in the field of computer networking. DISCUSS REST OF SLIDE

Here we will cover some of the basics of network security and will decide later if we want to cover Wifi or Security.. Poll.

Bad guys: put malware into hosts via Internet

- malware can get in host from:
  - *virus:* self-replicating infection by receiving/executing object (e.g., e-mail attachment)
  - *worm:* self-replicating infection by passively receiving object that gets itself executed
- spyware malware can record keystrokes, web sites visited, upload info to collection site
- infected host can be enrolled in botnet, used for spam. DDoS attacks

We attach devices to the Internet to get good stuff, but, unfortunately, along with all that good stuff comes malicious stuff—collectively known as **malware**—that can also enter and infect our devices. Once malware infects our device it can do all kinds of devious things, including deleting files; installing spyware etc. Our compromised host may also be enrolled in a network of thousands of similarly compromised devices, collectively known as a **botnet**, which the bad guys control and leverage for spam email distribution or distributed denial-of-service attacks against targeted hosts.

>Much of the malware out there today is **self-replicating**: once it infects one host, from that host it seeks entry into other hosts over the Internet, etc etc. In this manner, selfreplicating malware can spread exponentially fast. Malware can spread in the form of a virus or a worm.

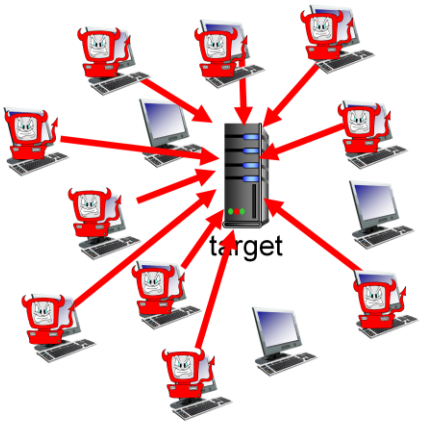
>**Viruses** are malware that require some form of user interaction to infect the user's device. The classic example is an e-mail attachment containing malicious executable code. If a user receives and opens such an attachment, BANG..self replicates by email

>**Worms** are malware that can enter a device without any explicit user interaction. For example, a user may be running a vulnerable network application to which an attacker can send malware. In some cases, without any user intervention, the application may accept the malware from the Internet and run it creating a worm. Worm scan inet for other hosts wit hsame vulnerabilioty and copies in that host.

Bad guys: attack server, network infrastructure

Denial of Service (DoS): attackers make resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic

1. select target
2. break into hosts around the network (see botnet)
3. send packets to target from compromised hosts



Introduction 1-66

Another broad class of security threats are known as **denial-of-service (DoS) attacks**. As the name suggests, a DoS attack renders a network, host, or other piece of infrastructure unusable by legitimate users. Web servers, e-mail servers, DNS servers (discussed next week), and institutional networks can all be subject to DoS attacks. Internet DoS attacks are extremely common, with thousands of DoS attacks occurring every year.

READ SLIDE

Most Internet DoS attacks fall into one of three categories:

>*Vulnerability attack.* This involves sending a few well-crafted messages to a vulnerable application or operating system running on a targeted host. If the right sequence of packets is sent to a vulnerable application or operating system, the service can stop or, worse, the host can crash.

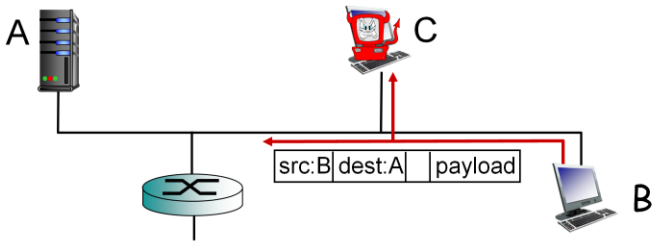
>*Bandwidth flooding.* The attacker sends a deluge of packets to the targeted host—so many packets that the target's access link becomes clogged, preventing legitimate packets from reaching the server.

>*Connection flooding.* The attacker establishes a large number of half-open or fully open TCP connections (TCP connections are discussed in 4 weeks) at the target host. The host can become so bogged down with these bogus connections that it stops accepting legitimate connections.

Bad guys can sniff packets

packet "sniffing":

- broadcast media (shared ethernet, wireless)
- promiscuous network interface reads/records all packets (e.g., including passwords!) passing by



- wireshark software used for end-of-chapter labs is a (free) packet-sniffer

Introduction 1-67

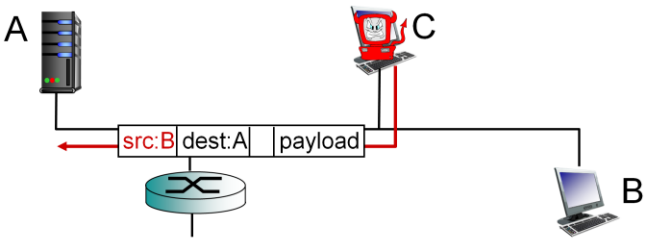
WiFi-connected laptops or smartphones are particularly vunerable to packet sniffers. While ubiquitous Internet access is extremely convenient and enables marvelous new applications for mobile users, it also creates a major security vulnerability—by placing a passive receiver in the vicinity of the wireless transmitter, that receiver can obtain a copy of every packet that is transmitted! A passive receiver that records a copy of every packet that flies by is called a **packet sniffer**.

Sniffers can be deployed in wired environments as well. In wired broadcast environments, as in many Ethernet LANs, a packet sniffer can obtain copies of broadcast packets sent over the LAN. Cable access technologies also broadcast packets and are thus vulnerable to sniffing. Furthermore, a bad guy who gains access to an institution's access router or access link to the Internet may be able to plant a sniffer that makes a copy of every packet going to/from the organization. Sniffed packets can then be analyzed offline for sensitive information.

Because packet sniffers are passive—that is, they do not inject packets into the channel—they are difficult to detect.

Bad guys can use fake addresses

IP spoofing: send packet with false source address



Hands up for security or wifi?

Introduction 1-68

It is surprisingly easy to create a packet with an arbitrary source address, packet content, and destination address and then transmit this hand-crafted packet into the Internet, which will dutifully forward the packet to its destination. Imagine the unsuspecting receiver (say an Internet router) who receives such a packet, takes the (false) source address as being truthful, and then performs some command embedded in the packet's contents (say modifies its forwarding table). The ability to inject packets into the Internet with a false source address is known as **IP spoofing**, and is but one of many ways in which one user can masquerade as another user.

To solve this problem, we will need *end-point authentication*, that is, a mechanism that will allow us to determine with certainty if a message originates from where we think it does.

Chapter 1: roadmap

1.1 what *is* the Internet?

1.2 network edge

- end systems, access networks, links

1.3 network core

- packet switching, circuit switching, network structure

1.4 delay, loss, throughput in networks

1.5 protocol layers, service models

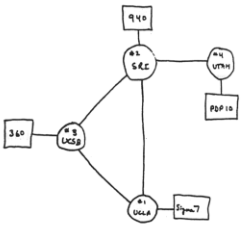
1.6 networks under attack: security

1.7 history

Internet history

1961-1972: Early packet-switching principles

- 1961: Kleinrock - queueing theory shows effectiveness of packet-switching
- 1964: Baran - packet-switching in military nets
- 1967: ARPAnet conceived by Advanced Research Projects Agency
- 1969: first ARPAnet node operational
- 1972:
  - ARPAnet public demo
  - NCP (Network Control Protocol) first host-host protocol
  - first e-mail program
  - ARPAnet has 15 nodes



THE ARPA NETWORK

Three research groups around the world, each unaware of the others' work began inventing packet switching as an efficient and robust alternative to circuit switching. The first published work on packet-switching techniques was that of Leonard Kleinrock [Kleinrock 1961; Kleinrock 1964], then a graduate student at MIT. Using queuing theory, Kleinrock's work elegantly demonstrated the effectiveness of the packet-switching approach for bursty traffic sources.

>In 1964, Paul Baran [Baran 1964] at the Rand Institute had begun investigating the use of packet switching for secure voice over military networks.

>Licklider and Roberts, both colleagues of Kleinrock's at MIT, went on to lead the computer science program at the Advanced Research Projects Agency (ARPA) in the United States. Roberts published an overall plan for the ARPAnet, the first packet-switched computer network and a direct ancestor of today's public Internet.

>In 1969, the first packet switch was installed at UCLA under Kleinrock's supervision, and three additional packet switches were installed shortly thereafter at the Stanford Research Institute (SRI), UC Santa Barbara, and the University of Utah. The fledgling precursor to the Internet was four nodes large by end of 1969.

>By 1972, ARPAnet had grown to approximately 15 nodes and was given its first public demonstration. The first host-to-host protocol between ARPAnet

end systems, known as the network-control protocol (NCP), was completed. With an end-to-end protocol available, applications could now be written. The first e-mail program in 1972.

Internet history

1972-1980: Internetworking, new and proprietary nets

- 1970: ALOHAnet satellite network in Hawaii
- 1974: Cerf and Kahn - architecture for interconnecting networks
- 1976: Ethernet at Xerox PARC
- late70's: proprietary architectures: DECnet, SNA, XNA
- 1979: ARPAnet has 200 nodes

Cerf and Kahn's internetworking principles:

- minimalism, autonomy - no internal changes required to interconnect networks
- best effort service model
- stateless routers
- decentralized control

define today's Internet architecture

In the early to mid-1970s, additional stand-alone packet-switching networks besides ARPAnet came into being: ALOHANet, a microwave network linking universities on the

Hawaiian islands – included the first multiple-access protocol using radio

The number of networks was growing. With perfect hindsight we can see that the time was ripe for developing an encompassing architecture for connecting networks together. Pioneering work on interconnecting networks (under the sponsorship of the DARPA), in essence creating a *network of networks*, was done by Vinton Cerf and Robert Kahn in 1974; the term *internetting* was coined to describe this work. These architectural principles were embodied in TCP (early TCP different from today's TCP). combined a reliable in-sequence delivery of data via end-system retransmission (still part of today's TCP) with forwarding functions (which today are performed by IP).

Metcalfe and Boggs built on Alohhas multiple-access protocol work when they developed the Ethernet protocol motivated by the need to connect multiple PCs, printers, and shared disks

READ REST OF SLIDE

Internet history

1980-1990: new protocols, a proliferation of networks

- 1983: deployment of TCP/IP
- 1982: smtp e-mail protocol defined
- 1983: DNS defined for name-to-IP-address translation
- 1985: ftp protocol defined
- 1988: TCP congestion control
- new national networks: Csnet, BITnet, NSFnet, Minitel
- 100,000 hosts connected to confederation of networks

By the end of the 1970s, approximately two hundred hosts were connected to the ARPAnet. By the end of the 1980s the number of hosts connected to the public Internet, a confederation of networks looking much like today's Internet, would reach a hundred thousand.

The 1980s would be a time of tremendous growth. Much of that growth resulted from several distinct efforts to create computer networks linking universities together. BITNET provided e-mail and file transfers among several universities in the Northeast. CSNET (computer science network) was formed to link university researchers who did not have access to ARPAnet. In 1986, NSFNET was created to provide access to NSF-sponsored supercomputing centers.

In the ARPAnet community, many of the final pieces of today's Internet architecture were falling into place.

>1983 saw the official deployment of TCP/IP as the new standard host protocol for ARPAnet (replacing the NCP protocol).

>1983 also saw the emergence of DNS, used to map between a human-readable Internet name (for example, www.mu.ie) and its 32-bit IP address, was also developed

READ REST OF SLIDE

Internet history

1990, 2000 ' s: commercialization, the Web, new apps

- early 1990' s: ARPAnet decommissioned
- 1991: NSF lifts restrictions on commercial use of NSFnet (decommissioned, 1995)
- early 1990s: Web
  - hypertext [Bush 1945, Nelson 1960' s]
  - HTML, HTTP: Berners-Lee
  - 1994: Mosaic, later Netscape
  - late 1990' s: commercialization of the Web

late 1990' s – 2000' s:

- more killer apps: instant messaging, P2P file sharing
- network security to forefront
- est. 50 million host, 100 million+ users
- backbone links running at Gbps

EXPLOSION

>ARPAnet, the progenitor of the Internet, ceased to exist.

>In 1991, NSFNET lifted its restrictions on the use of NSFNET for commercial purposes. NSFNET itself would be decommissioned in 1995, with Internet backbone traffic being carried by commercial Internet Service Providers.

>The main event of the 1990s was to be the emergence of the World Wide Web application, which brought the Internet into the homes and businesses of millions of people worldwide. The Web served as a platform for enabling and deploying hundreds of new applications that we take for granted today, Google, facebook…

>The Web was invented at CERN by Tim Berners-Lee between 1989 and 1991 based on ideas originating in earlier work on hypertext from the 1940s by Vannevar Bush and since the 1960s by Ted Nelson.

>Berners-Lee and his associates developed initial versions of HTML HTTP, a Web server, and a browser—the four key components of the Web.

>At about this time several researchers were developing Web browsers with GUI interfaces, including Marc Andreessen, who along with Jim Clark, formed Mosaic Communications,

which later became Netscape Communications Corporation

The second half of the 1990s was a period of tremendous growth and innovation for the Internet, with major corporations and thousands of startups creating Internet products and services. By the end of the millennium the Internet was supporting hundreds of popular applications, including four killer applications:

- E-mail, including attachments and Web-accessible e-mail
- The Web, including Web browsing and Internet commerce
- Instant messaging, with contact lists
- Peer-to-peer file sharing of MP3s, pioneered by Napster

Internet history

2005-present

- ~750 million hosts
  - Smartphones and tablets
- Aggressive deployment of broadband access
- Increasing ubiquity of high-speed wireless access
- Emergence of online social networks:
  - Facebook: soon one billion users
- Service providers (Google, Microsoft) create their own networks
  - Bypass Internet, providing "instantaneous" access to search, emai, etc.
- E-commerce, universities, enterprises running their services in "cloud" (eg, Amazon EC2)

JUST READ

Introduction: summary

covered a "ton" of material!

- Internet overview
- what's a protocol?
- network edge, core, access network
  - packet-switching versus circuit-switching
  - Internet structure
- performance: loss, delay, throughput
- layering, service models
- security
- history

you now have:

- context, overview, "feel" of networking
- more depth, detail *to follow!*