Overview

The course **Protocols and Communication Networks** deals with networking as it appears to professionals in the 21st century. The lectures combine theory and practical examples, with the goal of understanding the contemporary picture.

The course is divided into six chapters:

1. **Engineering Overview of Computer Networking**: Review of basic concepts in networking from the point of view of telecommunication engineering. The order of topics follows the OSI model from the physical layer up to the application layer. Unlike a course in Internet programming, the emphasis is on layers that belong to the operating system.
2. **Access Networks**: Access networks connect the user to the global Internet. To most users, "the network" refers to the immediate access network and the service provided by a remote server. In this chapter, we discuss the aspects of access networks relevant to users at various levels.
3. **IPv6**: Version 6 of the Internet Protocol (IP) was designed to solve various problems associated with version 4 and provides an essential basis for providing advanced networking services.
4. **Basics of Wireless Networking**: For over 25 years wireless networks are part of our day-to-day experience and the basis for mobile communication that has changed many patterns of behavior. Despite the simplicity of using mobile telephones and tablets, the technology behind mobile communication is more complex by several orders of magnitude. This chapter presents a simplified introduction to radio communication and the two principle forms of wireless mobile networking — WiFi and cellular. We will pay particular attention to the common features of the two contexts — the local and global networks.
5. **Congestion and Flow Control in TCP**: Congestion and flow control are basic and essential functions for reliable transport, and for most use cases are implemented in the TCP protocol. These functions add overhead to network operation and in many circumstances degrades the transmission throughput unnecessarily.
6. **TCP over Wireless**: The TCP protocol was designed many years before the introduction of WiFi and cellular networking, and its methods of congestion and flow control are based on assumptions relevant to wired transmission. In this chapter we discuss the implications of TCP for wireless and mobile networks.

The course material combines presentations with full-text written lectures. The presentations provide explanations, illustrations, and basic definitions in short accessible form. Viewing the slides is not a substitute for reading the lectures and doing the exercises. Attempting to learn from just the presentations is not recommended.

Recommendations for self-study

As seen on the syllabus, the course is designed according to weekly subjects and related exercises. An online course is based on the expectation that students at an advanced stage will be familiar with the methods of study that work best for them, can apply the self-discipline required to study at an even pace, and will submit the exercises on their due dates. However, experience shows that the temptation to put off the study of an online course is strong (in the worst case, until the very end of the semester). Giving in to this temptation does not lead to positive results.

At first glance, it appears that there may be **too many topics** to learn in one semester. The impression that no one can acquire full mastery of all the material in each presentation is entirely correct. And there is no expectation of full mastery. The expectation is similar to a course on Java programming, in which the emphasis is on presentation of a general framework and not on memorizing every syntactic structure in every package. After successfully completing the course, students will be able to discuss the significance of each subject and locate the details required for each discussion, in the course material or in external sources. By studying at an even pace and absorbing the material slowly, one comes to see that advanced networking involves a small number of basic ideas with many specific examples. The exercises and old exams provide a reasonable picture of the level of understanding expected in the course.

For each topic, it is recommended to go over the presentations with the lectures open, keeping an eye on the exercise intended to complete the presentation. Some of the topics are easy to grasp from the presentations alone, and in others it is important to read the associated text to understand the context and details.

If anything in the course remains unclear, feel free to email me at martin@hac.ac.il and I will make every effort to answer within 24 hours.

Presentation 1

Engineering Overview of Computer Networking

Engineering Overview of Computer Networking: Review of basic concepts in networking from the point of view of telecommunication engineering. The order of topics follows the OSI model from the physical layer up to the application layer. Unlike a course in Internet programming, the emphasis is on layers that belong to the operating system.

For additional explanations on the contents of this chapter can look at the material of the BSc course Computer Networking at: https://cs.hac.ac.il/staff/martin/Networks

---

Slide 2 What is Computer Networking?

Computer Networking is a sub-discipline of Computer Science that focuses on the logical distinction between local **computation** and **communication** between local computing agents. Real systems cannot be fully and clearly separated in this way, and the distinction should be understood as a formal abstraction.

In this model, **computation** includes the local operations of associated with a single CPU (central processing unit) — ALU operations, data loads and stores, branch instructions, OS (operating system) functions and so on. **Communication** refers to the exchange of coded information between computation units. The information exchanged may be data or instructions. The separation of processors into local and remote computation units depends upon the context. For example, two identical CPUs working concurrently on multi-threaded code within a dual core processor are said to share local data, but two identical CPUs working concurrently in a geographically distributed computing array are said to communicate.

For a given computing process, communication takes place between periods of local computation, involving 4 separate stages:

1. The local computing agent reaches a stage in its local operations that requires it to send an **information request** to a remote computing agent.
2. The remote computing agent **accepts** the information request and performs local computation (at the remote end) to produce the information.
3. The remote computing agent sends the requested information as a **response** to the accepted request.
4. The local computing agent receives the response from the remote computing agent.

Analysis of networking processes requires attention to the four stages as distinct events.

Top-Down versus Bottom-Up

Computer networking involves many complex interactions of qualitatively different types. Historically, networks were developed from the bottom up:

1. Simple systems were developed for transmitting coded information from point to point.
2. It became important to manage the information flows between transmitter and receiver.
3. It became possible to automatically switch information flows among different receivers.
4. Detailed management of multiple information exchanges came under software control.
5. Control of the end-to-end communication process came under control of user application programs.

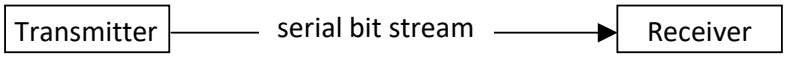
Telecommunications engineers study networking from the bottom up, in part because this is how networking infrastructure is designed and developed, and in part because that infrastructure is the locus of their expertise and responsibility. **Programmers** generally study networking from the top down, in part because this is where application software interacts with networking systems, and so the high level networking interfaces are the locus of their expertise and responsibility.

As a graduate course, our goal is not merely to review the basics of Internet communication, but to explore areas of contemporary R&D addressed to certain weakness of existing Internet technology. At the top of the application layer is the **Content Defined Network (CDN)**, which can be understood as an object-oriented network design in which application view objects defined in a universal name space and the network transparently locates and delivers data fragments that may be geographically distributed. CDNs are expected to operate smoothly on wireless mobile devices, providing such services as music and video streaming. However, the simple TCP/IP protocols at the base of Internet technology were designed 50 years ago and are not adequate to meet these expectations. In order to make these inadequacies clear, this course is organized from bottom up.

1. Chapter 1 prevents a bottom up overview of computer networking.
2. Chapter 2 discusses access networking (how we connect to the Internet) in order to make the material of Chapter 1 concrete.
3. Chapter 3 presents IP version 6 and discusses its relevance to emerging networking services.
4. Chapter 4 provides an overview of wireless networking — wireless local area networks and cellular networks.
5. Chapter 5 discusses congestion control mechanisms in widely used Internet protocols, which are required to permit the proper functioning of high volume streaming services.
6. Chapter 6 explores the problems associated with existing congestion control mechanisms operating over wireless networks and discusses proposals for updated protocols.
7. Chapter 7 presents video streaming and media session protocols associated with CDNs.

Slides 3 – 4 Physical Transmission

The basic element of data is the **bit** (binary digit) and the basic element of networking is the transmission of a single stream of serial data (sequential bits).



Serial data is characterized at the physical layer by its serial data rate in units of **bits per second**, written as bps or b/s.

1 byte is 8 bits and so

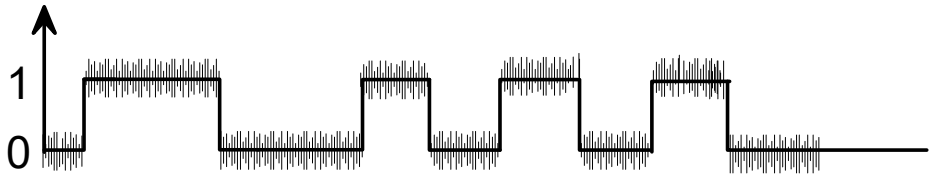
1 Byte per second = 1 B/s = 8 b/s

A data transmission rate of 30 MB/s (30 million bytes per second) is equivalent to 240 Mbps (240 million bits per second).

For data storage, 1 K = 2^{10} = 1024 $\Rightarrow$ 1 M = 1024^2 = 1,048,576 $\Rightarrow$ 1 G = 1024^3 = 1,073,741,824.

For data transmission, 1 k = 10^3 = 1,000 $\Rightarrow$ 1 M = 10^6 $\Rightarrow$ 1 G = 10^9

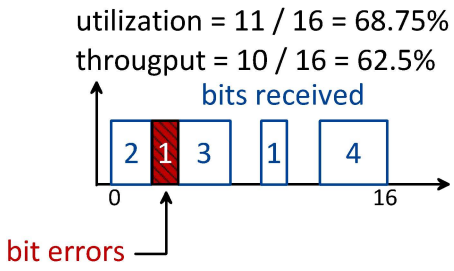
Capacity (bandwidth) is the maximum feasible data rate on a transmission medium. It is determined by the characteristics of the transmitter, the medium, and the receiver. The fundamental limits are the speed of the physical circuits and the **signal to noise ratio (SNR)** at the receiver. The receiver accepts the transmitted signal after its strength has diminished in the medium. In addition, other unwanted sources of input (**noise**) are added to the **desired signal**.



If the added noise is strong enough to change the weakened signal level from 0 to 1 or from 1 to 0 then this causes a bit error (the received bit is misread). As data rates get higher, it becomes more difficult to distinguish 0 from 1 in the presence of noise, and so data rates are limited by the **SNR**.

Utilization is the percent of capacity the transmitter attempts to use.

Throughput is the actual error-free data received per second divided by the capacity. The throughput is decreased if no data is sent (less than 100% utilization), if there are errors, or if data is delayed in an intermediate buffer.



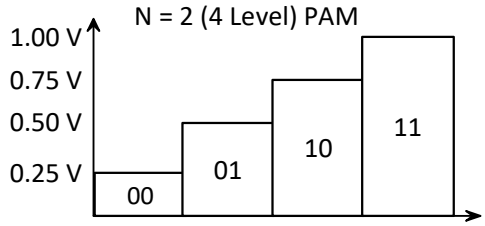
In the figure, the capacity in a certain transmission time is 16 bits — 11 bits are sent in 3 groups, so the utilization is 11/16, and only 10 bits are correctly received giving a throughput of 10/16.

Slides 5 – 6 Baud Rate

A symbol is a physical signal that encodes 1 or more bits. The **Baud rate** is the number of symbols transmitted per second.

For example, in **Pulse Amplitude Modulation (PAM)** the system permits 2^N different electrical levels which can be labeled by an N-bit binary number as 00...0 to 11…1. Each electrical level is a symbol that represents (transmits) N data bits (the binary numbering of the level).

In **4-level PAM**



We could define the signal as one of 0.25 Volts, 0.50 Volts, 0.75 Volts, or 1.00 Volt. The 4 levels are labeled 00, 01, 10, 11 — the 2-bit sequence received is the label of the measured level at the receiver.

A 33 kbps (33,000 bit per second) **dial-up modem** defines $2^{10} = 1024$ different electrical levels (combinations of amplitude and phase) which is the maximum number that can be distinguished in the presence of the usual electrical noise on a telephone line. The maximum transmission rate on a standard telephone line is 3300 Baud (symbols per second). The serial data rate is then

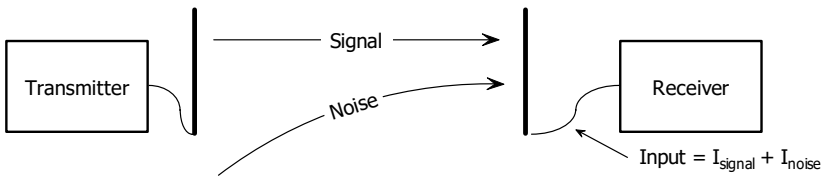
Data rate = (3300 symbols / second) $\times$ (10 bits / symbol) = 33,000 bps

Slides 7 – 14 Modeling Information — Signals and Noise

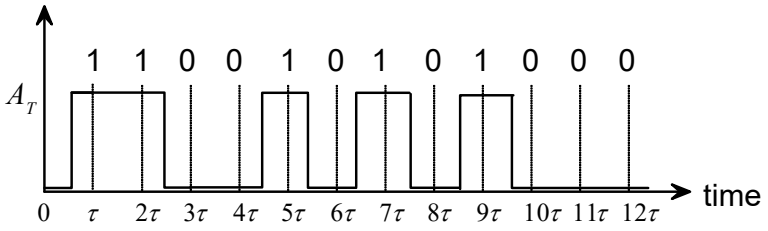
The word **information** is used to refer to a transmission that improves the likelihood of estimating the outcome of a test (the answers to a question). The universe of possible outcomes may be a finite set (yes/no, day of week, 256 pixel colors, etc) or an infinite set ("you won't believe what happened today!").

Communication is a transmission of symbols to a receiver. Before the transmission the receiver has limited knowledge of the symbol — only its permitted range and a statistical distribution of symbols within the range. After the transmission receiver has better (but not perfect) knowledge of the outcome (the symbol that was transmitted). The receiver tests the message to decides on the most likely symbol (outcome) but that decision is limited by noise.

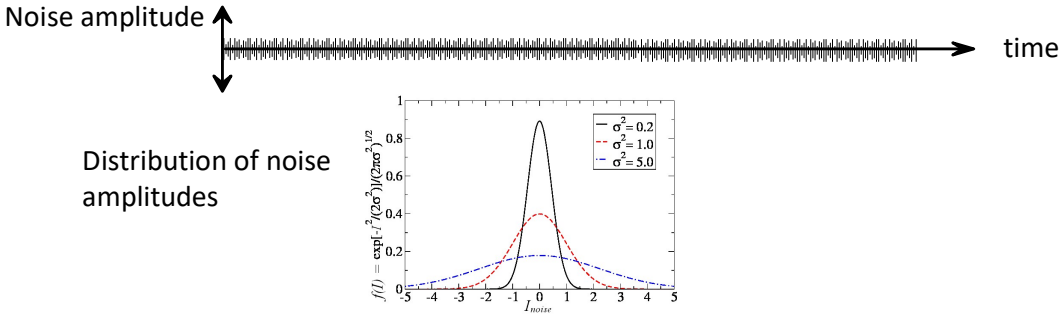
Sources of noise include interference from similar equipment, rounding-off errors, basic resolution of the detector, etc. Communication does not determine a unique outcome, but only an estimate subject to a probability of successful detection.



In a **2-level (binary) system** the transmitter sends a signal with level of 0 to represent a binary 0 and a level of A_T to represent a binary 1. When the signal reaches the receiver the maximum level has decreased to $A < A_T$.



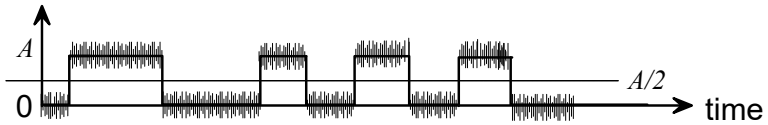
The noise added to the system this system can be characterized as "white noise" — a random current whose amplitude (level) obeys a Gaussian distribution ("bell curve").



The Gaussian distribution has mean and variance

$$\langle I_{noise} \rangle = 0 \qquad \langle {I_{noise}}^2 \rangle - \langle I_{noise} \rangle^2 = \langle {I_{noise}}^2 \rangle = \sigma^2$$

The received signal together with the noise is then



and the desired signal has and average and average squared of

$$\langle I_{signal} \rangle = \tfrac{1}{2} A \qquad \langle {I_{signal}}^2 \rangle = \tfrac{1}{2} A^2$$

The **Signal to Noise Ratio (SNR)** is defined as

$$SNR = \frac{\langle {I_{signal}}^2 \rangle}{\langle {I_{noise}}^2 \rangle} = \frac{A^2}{2\sigma^2}$$

The received must decide whether each symbol represents 0 or 1 and uses the rule

Signal < A / 2 $\Rightarrow$ binary 0
Signal > A / 2 $\Rightarrow$ binary 1

The probability of a bit error is the likelihood that the noise is large enough to make 0 larger than A/2 or make A smaller than A/2. The explicit calculation is shown on slide 10, and the probability depends only on the SNR. If SNR = 1 (equal signal and noise) the probability of an error is almost 25%. If SNR = 25 then the likelihood drops to about 2×10^{-4}.

A more useful formula is given by **Shannon's theorem**. We assume that there are $K = 2^k$ different symbols (a symbol is a k-bit binary integer). One symbol is then 1 out of K possible symbols — in other words, one symbol contains $k = \log_2 K$ bits.

The system transmits one symbol every τ seconds, so the Baud rate is $W = 1/\tau$. The bit rate is then

$b = (W \text{ symbols / second}) \times (k \text{ bits / symbol}) = W \times k \text{ bits / second} = W \log_2 K \text{ bps}$

Shannon's theorem states that the maximum size of the symbol set is given by

$$K_{maximum} = 1 + SNR$$

So that the maximum capacity on a transmission line is

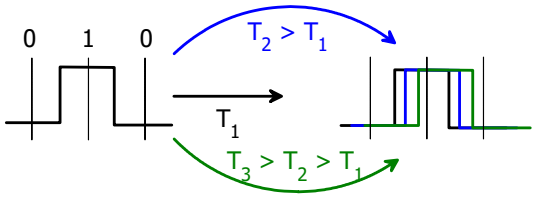
$$\text{Capacity} = W \times \log_2[1+SNR]$$

In order to achieve a transmission rate of 33 kbps on a dialup modem, the required SNR is shown on slide 12 to be 1023.

In very noisy environments some systems use **Direct Sequence Spread Spectrum (DSSS)** encoding. Here each binary bit (0 or 1) is transmitted as a coded sequence of m transitions called **chips** (m chips = 1 bit). Slide 13 shows an 8-chip code in which a binary 1 is transmitted as the chip sequence 10110100 and a binary 0 is transmitted as 01001011. In order to convert a 0 bit to a 1 bit requires errors in more than 4 chips. Slide 14 shows that is the probability of an error in a single chip is 1% = 10^{-2} then the likelihood or 4 chip errors in a single bit is about 10^{-6} (one in a million).

Slide 15 Inter-Symbol Interference

Sometimes, especially in wireless systems, a transmitted signal arrives by multiple paths with various time delays between transmission and reception. The receiver must try to make sense of the signal arriving at various start times.



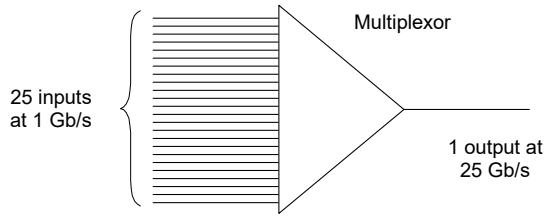
The receiver sees **inter-symbol interference (ISI)** — the interference between one signal arriving more than once at different times. If the ISI is not too bad, the receiver calculates the optimum time to sample the signals — at the time that they all agree.

If the delay between signals varies over time (from one transmitted bit to the next), then the optimum sampling time is difficult to compute — this is called **jitter**.

Slides 16 – 20 Data Concentration

Generally, no single node can utilize the capacity on a high capacity link. For example, an optical fiber cable containing 4 fibers each operating at 25 Gbps has a capacity of 100 Gbps. Very few sources can produce a data stream at this rate over an extended period of time.

High capacity links often carry the output of a data concentrator that combines the output data streams from multiple nodes into one very fast data stream. For example, a single optical fiber operating at 25 Gbps can transmit the combined outputs of 25 nodes each producing a 1 Gbps data stream.

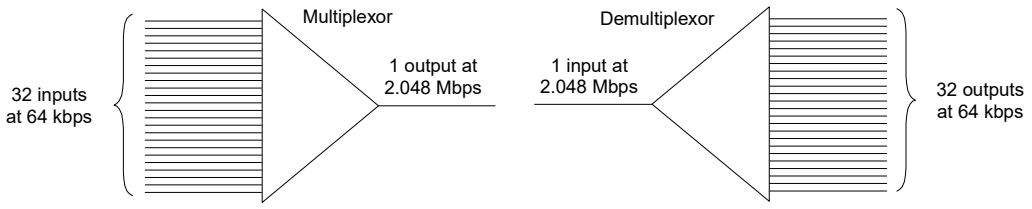


Combining data streams into one aggregate data streams is called **multiplexing**. Two basic methods of multiplexing are **Frequency Division Multiplexing (FDM)** and **Time Division Multiplexing (TDM)**.

Frequency Division Multiplexing (FDM) divides the available transmission frequencies (bandwidth) among multiple nodes. Each node is assigned a different frequency band and all nodes can transmit simultaneously. For example, commercial FM radio uses the 20 MHz frequency band from 88 MHz to 108 MHz. By dividing the 20 MHz band into 100 channels, each FM channel is allotted a 200 kHz band. The first channel is assigned a center frequency of 88 MHz and transmits in the band 88 MHz $\pm$ 100 kHz = 87.9 to 88.1 MHz.

The **required bandwidth** for signal transmission depends on the signal content. For example, the human ear can generally hear audio frequencies in the range 0 to 25 kHz. The details of FM radio require a bandwidth 4 times this range, so that a minimum band of 100 kHz is required. The required bandwidth for a **digital signal** depends on the transmission method, so that a data stream at 1 Mbps usually requires a frequency band between 750 kHz and 2 MHz. By dividing the available frequencies into channels, commercial FM radio permits 100 stations to transmit simultaneously at 1/100 of the maximum transmission rate available to the whole 20 MHz band.

Time Division Multiplexing (TDM) divides the capacity into time slots so that each node transmits at full capacity but only during its assigned time slot. For example, an E1 digital line (used in telephone service) transmits at 2048 kbps. This total capacity is divided into 32 time slots. The data from each node is transmitted at 2048 kbps for 1/32 of the time, leading to an average transmission rate of 64 kbps per node (32 x 64 kbps = 2048 kbps).



Slide 19 shows the E1 multiplex system used in telephone systems. Each user telephone produces a 64 kbps bit stream, produced by sampling the voice channel 8000 times per second and representing the voice sample by 1 byte = 8 bits.

Each sample captures 1/(8000 samples per second) = 125 microseconds per voice sample. The multiplexor (MUX) receives 1-byte samples from 32 channels simultaneously (in parallel) every 125 μsec and transmits each byte in only 3.91 μsec = 125 μsec / 32. This way, the MUX transmits all 32 1-byte samples at 32 × 64 kbps = 2048 kbps in exactly 125 μsec. The MUX is then ready to accept the next 1-byte samples from 32 channels.

The collection of 32 1-byte samples is called a **data frame**. The sampling mechanism in E1 is called **isochronous** ("equal times") because each sample follows the previous after a fixed time interval.

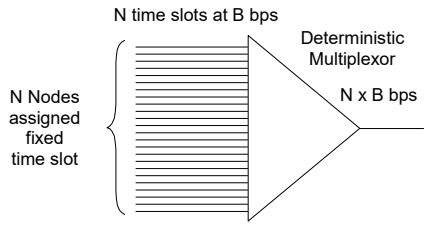
The **GSM** system for 2G and 3G cellular telephone systems uses a form of mixed multiplexing, combining FDM and TDM. The GSM radio frequency band of 25 MHz is divides into 125 channels with a bandwidth of 200 kHz per channel that is used to transmit data at a rate of 270 kbps. The 270 kbps data stream per channel is divided into 8 times slots and each user can transmit at 33 kbps.

Slides 21 – 24 Data Statistics

Most data streams can be characterized as **Constant Bit Rate (CBR)** or **Variable Bit Rate (VBR)**.

Constant Bit Rate (CBR) streams carry isochronous data produced at an equal time interval between bits. The bit rate (bits per second) is constant and identical to the average data rate, the peak data rate and the minimum data rate. An example is uncompressed digital audio, in which the digital system samples the analog signal every T seconds, rounds off the sample to an n-bit number, and produces a digital audio stream at (n/T) bps.

CBR data streams require **deterministic multiplexing**. The data streams from N nodes are assigned N reserved time slots with guaranteed transmission capacity. Each node transmits in its assigned time slot.



If each node transmits at B bps then the multiplexor and output line must have capacity for a data stream of N×B bps.

In the telephone system, two hierarchical systems of deterministic multiplexing exist — the **PDH** and the **SDH**. In the PDH, an E1 line carries 32 digital streams of 64 bps, and E2 line carries 4 multiplexed E1 lines and so on. In addition to multiplexing, the digital transmission lines can be leased from telephone companies for other transmission purposes.

Variable Bit Rate (VBR) streams carry data that appears as random bursts of bits transmitted at some maximum data rate. The **peak data rate B** is larger than the **average data rate λ** (which takes into account times with no transmissions).



To simplify statistical analysis we generally assume that data packets are mutually independent, so that Poisson statistics apply. Thus, we do not know when bits will arrive at the receiver but we can use the **Poisson distribution**

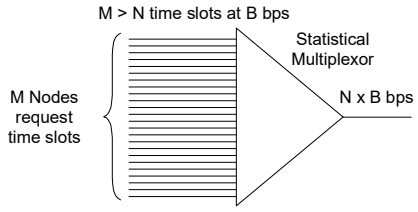
$P(k,T,\lambda)$ = probability of k bits arriving during T seconds when the average number of bits is λT

$$P(k,T,\lambda) = \frac{(\lambda T)^k}{k!} e^{-\lambda T}$$

For example, the data sent by a time-of-day client request is 1000 bits. The request is sent once every hour (3600 seconds), so the average data rate is 1000 bits / 3600 seconds = 0.28 bps. The request is transmitted at full speed over 802.11g WiFi at a peak data rate of 55 Mbps. So, the peak data rate B = 55 Mbps > average data rate λ = 0.28 bps. The likelihood of receiving 1 bit during any interval of 1 second is

$$P(1 \text{ bit}, 1 \text{ second}, \lambda = 0.28) = \frac{(0.28)^1}{1!} e^{-0.28} = 0.21$$

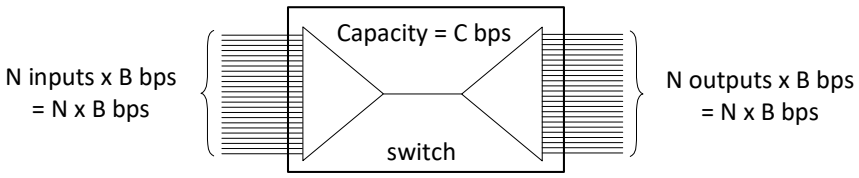
In order to lower the cost of high capacity systems, VBR data streams can be multiplexed using **statistical multiplexing**. A system serving M nodes provides N time slots, where M>N. The average data rate λ is less than the peak data rate B and so the total capacity = N x B can be set to a value larger than the average traffic = M x λ but smaller than the maximum capacity M x B.



As long as the actual traffic is less than the capacity the system works well. If the actual traffic is greater than the capacity then data will be delayed or lost. The most well-known example is Internet routers — when actual traffic is less that capacity data flows without delay, but when actual traffic increases above capacity (a condition called **network congestion**) data is delayed and throughput is slow.

Slides 25 – 27 Switching

A **data switch** is a combination of a multiplexor and Demultiplexor that permits data entering at input port i to exit at output port j, for i,j = 0, 1, 2, ... , N – 1.



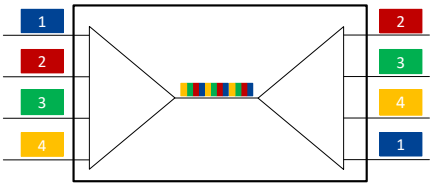
In this example

Data from node 1 is sent to node 2

Data from node 2 is sent to node 3

Data from node 3 is sent to node 4

Data from node 4 is sent to node 1



Switching is divided into two basic types, **Circuit Switching** and **Packet Switching**

Circuit switching (a **circuit mode connection**) uses deterministic multiplexing for CBR data streams, so that the capacity is $C = N \times B$, providing a dedicated (reserved) link between input port i and output port j. There is no competition for bandwidth and each input node is guaranteed capacity B, whether it is used or not. The best-known example is a Bezeq phone call in which the telephone switching network provides a dedicated 64 kbps link from telephone to telephone, even if no one speaks.

Packet switching (a **packet mode connection**) uses statistical multiplexing to switch VBR data with capacity $C = M \times B < N \times B$. The time slots are assigned dynamically on request, so that data from input port i is transmitted to output port j unless competition creates congestion. For example, an Internet router contains a packet queue from which data is transferred on a first come first served basis.

Slides 28 – 29 Message Delay

Message delay is the time between the transmission of the first bit of a message and the receipt of the last bit of the message. There are 4 basic contributions:

Transmission delay T_T is the time required to inject the message bits into the line:

T_T = (bits in packet) / (bits per second)

Example: 1000 Mbits / 100 Mbps = 10 sec

Processing delay T_{proc} is the packet processing time in intermediate nodes, which depends on the complexity of the routing and forwarding operation.

Propagation delay T_{prop} is the time required for the signals to reach the receiver from the transmitter.

T_{prop} = (length of cable) / (signal speed)

Except in satellite communication, this time is usually negligible. For example, the propagation delay over 4 km of fiberoptic cable is

4 km / (2×10^8 km/s) = 2×10^{-8} sec which is much smaller than T_T = 10 seconds

Queuing delay T_Q is the time packet waits in a buffer for previous packets to be processed in conditions of congestion.

T_Q = (service time per packet) × (packets waiting in buffer + 1)

A general result of queuing theory is that the average number of packets waiting in a buffer is given by

Buffer level = utilization / (1 – utilization)

Buffer level + 1 = utilization / (1 – utilization) + 1 = 1 / (1 – utilization)

For example, suppose that the average service time per packet is 10 ms / packet, so that the average service rate is 1/(average service time) = 100 packets / second. Suppose also that the average traffic rate is 85 packets / second, so that

Utilization = (85 packets / second) / (100 packets / second) = 0.85

Buffer level = 0.85 / (1 – 0.85) = 5.7 packets waiting for service

T_Q = (10 ms / packet) / (1 – 0.85) = 67 ms

Suppose that the switch capacity is C = 100 packets / second. If actual traffic (demand) is greater than 100 packets / second then the buffer will overflow causing excess delay. The likelihood that that the demand will exceed C is given by

$$
\begin{aligned}
P(\text{demand } > C) &= P(\text{demand} = C+1) \text{ OR } P(\text{demand} = C+2) \text{ OR } P(\text{demand} = C+3) \text{ OR ...} \\
&= \sum_{k=C+1}^{\infty} P(\text{demand } = k) \\
&= \sum_{k=C+1}^{\infty} \frac{(\text{average demand})^k}{k!} e^{-(\text{average demand})} \\
&= \sum_{k=101}^{\infty} \frac{85^k}{k!} e^{-85} \\
&= 0.05
\end{aligned}
$$

There is a 5% probability of buffer overflow.

Slide 30 Error Control

A **bit error** is when a data 1 is received as 0 or a data 0 received as 1.

$$\text{Bit Error Rate (BER)} = \frac{\text{bit errors in received data}}{\text{bits in received data}}$$

Packet loss is when congestion or buffer overflow causes a packet to be discarded (deleted).

$$\text{Packet loss rate} = \frac{\text{packets lost}}{\text{packets transmitted}}$$

Error detection is performed by transmitting a hash function of the data along with the data itself. The hash is a simple checksum, a redundancy code, or an error correction code. The receiver compares its own independent calculation of the hash with the transmitted code to determine if there are errors in the data (or perhaps in the transmitted hash).

Error control requires that any corrupt packet must be discarded. Optionally the receiver may request that the discarded / missing packets be retransmitted.

Slides 31 – 33 Flow Control and Congestion Control

Flow control is a system by which the sender avoids causing overflow in the receiver's buffer.

Congestion control is a system by which all senders avoid causing overflow of buffers in intermediate network devices (routers, switches, etc).

Characteristics of buffer operation include:

Buffer arrival rate is the number of bytes / second arriving from network

Buffer empty rate is the number of bytes / second leaving the buffer to the network or to the application requesting the data.

Buffer file time is $T_{overflow} = \frac{\text{buffer size}}{\text{buffer arrival rate} - \text{buffer empty rate}}$

For example a 64 KB buffer filling at 8 KB/sec and emptying at 4 KB/sec will overflow after

$$T_{overflow} = \frac{64 \text{ KB}}{8 \text{ KB/sec} - 4 \text{ KB/sec}} = \frac{64 \text{ KB}}{4 \text{ KB/sec}} = 16 \text{ seconds}$$

Queuing theory is a general mathematical theory of buffer queues. Under the following assumptions:

- Data packets arrive at the buffer independently and obey Poisson statistics
- The packets are of random length in bytes
- There is a fixed average packet arrival rate in the steady state (long time average)
- Data packets leave the independently and obey Poisson statistics
- There is a fixed average packet emptying rate in the steady state

The following results are obtained:

$$\rho = \text{Utilization} = \frac{\text{arrival rate}}{\text{empty rate}}$$

$$\text{Latency} = \text{buffer queuing delay} = \frac{1}{\text{empty rate} - \text{arrival rate}} = \frac{1}{\text{empty rate}}\left(\frac{1}{1-\rho}\right)$$

$$\text{Buffer Level} = \text{Latency} \times \text{arrival rate} = \frac{\rho}{1-\rho}$$

As shown in the graph on slide 32, latency (queuing delay) and buffer level rise very quickly above a utilization of about 85%.

A **highly oversimplified model** of buffer throughput is that throughout (packets received from the buffered transmission system relative to the system maximum) rises linearly with utilization up to the maximum capacity. If the line utilization is 90% we expect that the system throughput is also 90%. Above the system capacity, no additional packets can be handled so the throughput levels off at 100%. This model is not realistic.

Realistic throughput behavior reflects several complications: When there is a high arrival rate at the buffer there is longer latency and possibly overflow. As packets are discarded or delayed, the receiver or sender reaches a timeout stage — the packet is declared lost and is automatically retransmitted. These retransmissions add more data segments leading to an even higher arrival rate at the buffer, leading to more congestion. In practical systems, measured throughput begins to drop when utilization passes the 85% level.

Slide 34 Data Link

The earliest electronic communication systems (telegraph and telephone) consisted of point-to-point circuits between two dedicated devices. As these systems became more complex, additional functions were added to manage the communications channel. In today's networking devices, management is a primary task — a rule of thumb is that delivering data messages requires about 10% of the effort and making the hardware work correctly requires about 90% of the effort.

Hardware management is often called **Operations, Administration, and Maintenance (OAM)** and permits the user application to assume that transmission infrastructure "just works." OAM activities include management of reliability, availability, stability, serviceability, and growth.

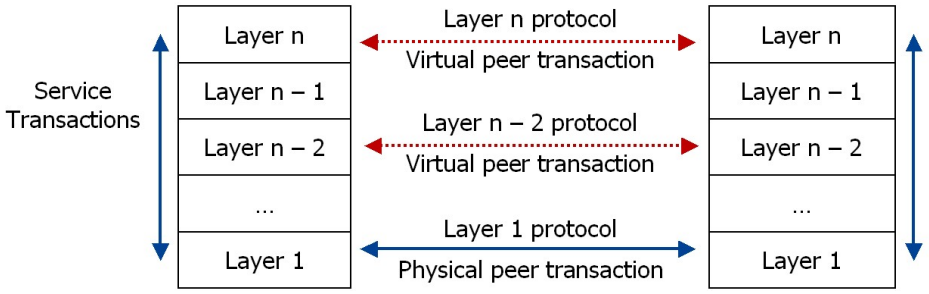
The division of activities in the communication hardware between transmission and management led to the concept of **layered protocols**. Transmission is defined at the **Physical Layer (PHY)** and management is defined at the **Data Link Layer** that supervises the PHY from above.

Slides 35 – 39 Layered Protocol Model

In a general layered communication model, the communication task is divided into layers. At each layer, a hardware or software entity or **agent** is defined that performs specific tasks to enable communication. The entities performing related tasks at the same layer are called **peers**. A specific **peer-to-peer protocol** defined between software/hardware entities acting at each layer. Viewing the layers in one peer the assembled layers form a **Protocol Stack**.

The layer n protocol is said to perform **virtual communication** between **peer entities** at layer n. A layer n protocol operates by:

- Accepting a service request from layer n + 1
- Processing information defined at layer n
- Passing a service request to layer n – 1
- Receiving responses from layer n – 1
- Processing layer n information
- Passing a response to layer n + 1

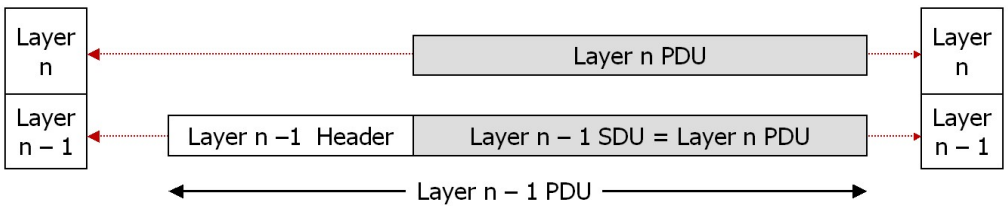


Protocol information at layer n is transmitted to the layer n peer in a **header** added to the message, in a process called **encapsulation**.

The layer n – 1 protocol receives a service request from layer n — this request is a message for the layer n peer agent. The protocol adds a layer n – 1 **header**, which is a message to the layer n – 1 peer agent.

A **Service Data Unit (SDU)** at layer n – 1 is the message received from layer n as part of a service request. The layer n – 1 protocol treats the SDU meaningless data to be carried by layer n – 1.

A **Protocol Data Unit (PDU)** at layer n – 1 is the message sent by the layer n – 1 protocol to its peer. The layer n – 1 PDU consists of a layer n – 1 header and the layer n – 1 SDU (which is just the layer n PDU).



Protocol stacks exist for many different communication systems, including Internet protocols, cellular protocols, and specific protocol stacks for large mainframe computing systems. The **Open System Interconnection Model (OSI)** proposed in the early 1980s was based on the IBM System Network Architecture (SNA) and few other systems are based on this model. However, OSI presents a very useful reference model for layering the overall communication task in terms of basic subtasks. OSI remains in use as a description of these tasks.

Layer	Function	Description
7	Application	Exchange of data between user applications in local and remote nodes
6	Presentation	Defines the syntax and semantics of exchanged data
5	Session	Responsible for the identification, separation, and continuity of multiple conversations (ongoing data transactions between software agents)
4	Transport	Responsible for reliable end-to-end data exchange between host nodes, preventing data loss, data errors, packet repetition, ordering errors
3	Network	Responsible for end-to-end routing of data between local and remote nodes over multiple hops
2	Data Link	Control of data transmission between neighboring hardware agents (one hop)
1	Physical	Data transmission between neighboring hardware agents on physical channels (electrical, optical, radio, …)

The **Internet protocol model** consists of 4 layers. These layers can be mapped to the OSI model in the following way:

<table>
<tr><th>OSI Layer</th><th>OSI Function</th><th>Internet Layer</th><th>Comment</th></tr>
<tr><td>7</td><td>Application</td><td rowspan="2">Application</td><td rowspan="2">Internet applications provide presentation service and some session services (for example HTTP transactions)</td></tr>
<tr><td>6</td><td>Presentation</td></tr>
<tr><td rowspan="2">5</td><td rowspan="2">Session</td><td rowspan="2">Transport</td><td rowspan="2">Internet session management can be:
Reliable — with OSI transport service
Unreliable — no OSI transport service</td></tr>
<tr><td>4</td><td>Transport</td></tr>
<tr><td>3</td><td>Network</td><td>Network</td><td>End-to-end data routing as in OSI</td></tr>
<tr><td>2</td><td>Data Link</td><td rowspan="2">Infrastructure</td><td rowspan="2">Internet protocols do not discuss physical data transmission</td></tr>
<tr><td>1</td><td>Physical</td></tr>
</table>

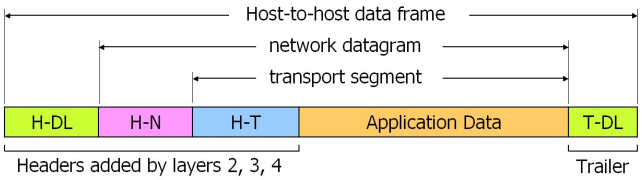
PDUs in the Internet protocols have specific names and address types:

Layer	Message	PDU Name	PDU Address for Sending/Receiving
Application	Data	Message	Socket defined by the local system's OS
Transport	Header	Segment	Port number assigned to client/server software by the local system's OS
Network	Header	Datagram	Network Address — IP address assigned by system administrator
Data Link	Header + Trailer	Frame	Hardware Address — MAC address assigned by manufacturer
Physical	Bits	Signal	Attachment — physical connection to device (cable or antenna)

Applications identify a peer application by its **endpoint** — the host **IP address** and the **port number** of the peer software agent. Port numbers are 16 bits in length. Port numbers from 0 to 1023 are reserved for specific **services** so that a client may locate a server by the IP address of the host and the **well-known port** number of the required service. Client software identifies itself by choosing a port number above 1024 from a pool of unassigned port numbers in the OS.

In the Internet model

- An application **message** is passed to the transport layer, which adds a transport header H-T to form a **segment**.
- The transport layer forwards the segment to the network layer, which adds a network header H-N to form a **datagram**.
- The network layer forwards the datagram to the data link layer, which adds a data link header H-DL and a trailer T-DL to form a **frame**.
- The data link layer forwards the frame to the physical layer, which transmits the frame as a bit stream.



Slides 40 – 41 Data Link Sublayers

Most common data link protocols are divided into two sublayers, **Logical Link Control (LLC)** and **Medium Access (MAC)**. In many systems (such as Ethernet and WiFi), MAC is mandatory and LLC is optional.

The **LLC** sublayer is responsible for multiplexing of multiple data sources and destinations, identification of the packet type, error correction and flow control. LLC controls reliability between neighboring nodes in much the same way that the transport layer controls reliability between hosts end-to-end.

The **MAC** sublayer is responsible for managing the network topology, medium access (how nodes transmit), sharing the transmission medium among nodes, permission to transmit at a given time, conforming to the data frame structure, hardware (MAC) addressing, and error detection.

Slides 42 – 43 Medium Access Sharing

Similar to data concentration, the sharing of the physical transmission medium among multiple hardware agents is based on division into time slots, frequency bands, or coding.

In **time division**, each host is granted the full bandwidth in an allocated time slot. The time slot may be allocated deterministically (as in telephone switching) or statistically (as in Ethernet and WiFi).

In **frequency division**, each host is granted partial bandwidth in all time slots. Examples include commercial radio / TV and Bluetooth.

In **code division**, each host granted full bandwidth in all time slots. Since each host transmits using different coding scheme, the transmissions can be separated. The main example is cellular Code Division Multiple Access (CDMA).

Some common shared medium networks are

Ethernet	IEEE 802.3 local area network Wired time-division access
WiFi	IEEE 802.11 local area network Wireless time-division access
Bluetooth	IEEE 802.15 personal area network Wireless frequency-division access
WiMAX	IEEE 802.16 metropolitan area network Wireless time/frequency-division access
GSM / UTMS	ITU 2G / 3G cellular network Wireless time/frequency-division access
CDMA / CDMA2000 / LTE	ITU 2G / 3G cellular network Wireless code-division access

Slide 44 Connection Types

A **connection** is a state machine associated with data exchange, which:

- Sets-up the channel before data is exchanged
- Monitors the channel state during data exchange
- Manages multiple transactions associated with the connection state
- Closes the channel after data exchange has completed

An example of a **connection-oriented** network protocol is a standard telephone call. The connection is made at the application layer when the user enters a phone number and the callee answers the call. After an extended conversation the caller or callee disconnects the connection.

An example of a **connectionless** network protocol (no associated state machine) is an email message. There is no connection at the application layer. The user sends the email, hopes that the message arrives, hopes that the message is found, and hopes that the message is read. The only guarantee that an email arrived is that the recipient replies with an independent email.

Protocols at any layer may be connection-oriented or connectionless. The details of the state machine usually make connection-oriented protocols more complex.

Slides 45 – 48 High-Level Data Link Control (HDLC)

HDLC is a family of data link protocols based on IBM's SDLC, the layer 2 protocol used in its mainframe SNA protocol stack. It was originally designed for communication between CPUs and peripheral equipment. The **Link Access Protocol (LAP)** family is a number of versions of HDLC used in public network architectures, most importantly the IEEE 802.2 LLC protocol used with Ethernet and WiFi, and PPP used in the Internet.

The general HDLC frame has the form

01111110	Address	Control	Data	CRC	01111110
8	8	8	≥ 0	16 / 32	8

Where

- 01111110 identifies the start and end of the frame
- The 8 bit address identifies up to 256 different hardware agents
- The control field specifies the type and function of the frame with the following meanings:

Information

7	6 5 4	3	2 1 0
0	SEQ N(S)	p/f	NEXT N(R)

Supervisory

7	6	5 4	3	2 1 0
1	0	type	p/f	NEXT N(R)

Unnumbered (connectionless)

7	6	5 4	3	2 1 0
1	1	type	p/f	subtype

- SEQ — a sequence number for maintaining the correct order of frames in a connection-oriented data link.
- NEXT — an acknowledgement (ACK) indicating the number of the next expected frame
- type — the type of ACK in a supervisory frame:
  00 = ACK received frame and ready to receive more frames
  01 = reject = last packet was corrupted — resend all frames starting from NEXT N(R)
  10 = ACK but not ready to receive (buffer full) — stop sending until type = 00
  11 = selective reject (resend packet with NEXT value)
- p/f — used for polling systems

Flow control in HDLC

Each **information frame** carries a 3-bit SEQ to number the frame modulo 8. The receiver uses the SEQ to insure that frames arrive in order, without missing or repeated frames. To acknowledge receipt of SEQ N(S) the receiver can send a **supervisory frame** where the NEXT field contains N(R) = SEQ N(S) + 1. For example, a sender information frame with SEQ = 4 is ACKed by a receiver supervisory frame with NEXT = 5.

If both sides are sending data then the NEXT can be sent in the information frame that carries the data. For example, station 1 sends data in an information frame with SEQ = 2 and NEXT = 1, and station 2 sends data in an information frame with SEQ = 1 (the frame expected by station 1) and NEXT = 3 (ACK of the frame with SEQ = 2 from station 1).

The **type field** in supervisory frames is used to control the flow of frames.

The **Point-to-Point Protocol (PPP)** is a layer 2 protocol commonly used between Internet routers, and between a host and its **Internet Service Provider (ISP)**. The frame structure is

01111110	11111111	11000000	Protocol	Data	CRC	01111110
8	8	8	8 or 16	≥ 0	16 / 32	8

For which,

- Address is 11111111, which means broadcast to all listeners
- HDLC control byte is 11000000, which indicates unnumbered data packets of default type in a connectionless data link
- The Protocol data field identifies a network protocol (such as IP) or a link negotiation protocol (another 2 sublayer).

Control sublayers include **Link Control Protocol (LCP)** and **Network Control Protocol (NCP)**.

LCP defines PPP options such as **header compression** (omit control / address fields), size of protocol / CRC fields, and data length, as well as testing and session termination.

NCP defines network layer options, such as network protocol, addressing, header compression (encoding of header fields), and user authentication (exchange of user name and password to ISP).

Slides 49 – 50 Ethernet

Ethernet is the most commonly used wired **local area network (LAN)**. In most modern implementations, a cable connects a network interface card (NIC) in the host to **switch** (hub) that forwards data link frames between physically connected devices.

The Ethernet frame has the following structure:

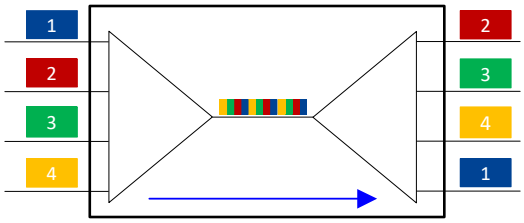
Preamble	**Start**	**Destination Address**	**Source Address**	**Type or Length**	**Data**	**CRC**
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46 – 1500 bytes	4 bytes

Preamble	7 bytes of 10101010 for synchronizing receivers	Header fields
Start	10101011	
Address	Hardware (MAC) address of node 48-bit fixed MAC addresses assigned by manufacturer Broadcast address FF:FF:FF:FF:FF:FF (frame read by all)	
Length	Length of data field in bytes (up to 1500)	
Type	Code identifying protocol of SDU in data field Used in most Ethernet systems Type codes > 1536 =0x600 (IP data is code 0x0800)	
CRC	CRC-32	Trailer

Ethernet is a simple network defined at the physical level (data rate, and other physical characteristics) and connectionless data link protocol at the MAC level.

Ethernet (**MAC**) addresses are 6 bytes (48 bits). Since the MAC protocol is connectionless, each frame contains the address of the **destination** that must identify frames addressed to it, and the **source** address so the destination knows which host is sending it data.

A switched Ethernet hub permits non-blocking transmission from any host at any time.



The ports on the left are outputs from hosts 1 – 4 and the ports on the right are inputs to the same hosts. In the example, the switch is sending frames from host 1 to host 2, host 2 to host 3, and so on. Simple Ethernet switches operate at 100 Mbps – 1 Gbps, and 10 Gbps (and faster) is available at higher cost.

At initialization, the switch enters **learning mode** (promiscuous mode). It does not know which physical port is connected to the host with a particular MAC address and so it must forward the frame to all 4 ports. If port 1 is connected to a host with MAC address 11:22:33:44:55:66 then when this host transmits the switch associates this MAC address with this port. The switch sends the frame to ports 2 – 4, but the addressed host reads the data. When this host replies to host 1 then the switch learns the port to MAC association for this host, and so on.

Slides 51 – 56 Asynchronous Transfer Mode (ATM)

ATM is a complex packet switch infrastructure defined at the physical and data link layers providing **connection-oriented** service. It is widely used in telephone infrastructure and for high performance private data networks.

Before data is exchanged, the system sets up an end-to-end connection between host addresses. Each host-to-host connection is identified by an 8-bit **Virtual Path Identifier (VPI)** and a 16-bit **Virtual Circuit Identifier (VCI)** so that data frames are forwarded according the VPI:VCI number. It is not necessary to include the 48-bit source and destination addresses. The switch is configured for fast data transfer at set-up time, so that no learning mode is necessary.

ATM switches operate up to 2.5 Gbps. The date frames, called cells, are small (53 bytes) to enable fast priority switching and low delay. The system defines complex **service parameters** (see slide 53) that must be guaranteed to each user. These parameters are designed to serve **four categories of data traffic** (see slide 54). Slides 55 and 56 illustrate some of the protocols and frame types defined in ATM. The details of ATM are far beyond the scope of this course — it is presented here as an example of a complex switching infrastructure defined at the physical and data link layers.

Slides 57 – 60 Connection-Oriented and Connectionless Routing

Routing packets between hosts, end-to-end over multiple hops and through multiple intermediate devices, is the main goal of the **network layer** (layer 3).

Connection-oriented routing defines a **virtual circuit (VC)**, so-called because it provides the service associated with a physical electronic circuit. In a **switched virtual circuit (SVC)** set-up and close messages carrying source and destination addresses are used to reserve a connection from host to host.

For example, slide 57 shows a set-up message to establish the path B $\rightarrow$ 1 $\rightarrow$ 4 $\rightarrow$ 6 $\rightarrow$ F. This path is labeled VC-1 and so data packets from B to F are identified by VC-1.

To increase flexibility, the VC label may be changed on each hop. The example on slide 58 shows that the path from node B to node C has the labels 14 (between B and switch 1), 23 (between switch 1 and switch 2), and 34 (between switch 2 and host C). The **ATM VPI:VCI** fields are examples of virtual circuit labels.

At switch 1, interface 1 is physically connected to node B, interface 2 is physically connected to node A, and interface 3 is physically connected to switch 2.

The intermediate switches 1, 4, and 6 hold **switching tables** used for forwarding frames. Switch 1 must know that an frame entering on interface 1 with VC=14 must be forwarded to interface 3 with VC=23. These switching tables can be implemented quickly to minimize processing delay T_{proc} in the switch.

Connection-oriented virtual circuit routing may use either **deterministic** multiplexing (circuit mode switching) or **statistical** multiplexing (packet mode switching).

Connectionless routing (also called **datagram service**) is used in the Internet. Because there is no set-up before data transfer, the system cannot reserve capacity for any given source and must always use **statistical** multiplexing (packet mode switching). Each datagram is routed individually through the network and so must carry the source and destination addresses in the header of every packet. Since end-to-end paths are not reserved in advance datagrams may follow separate routes, which may be an advantage if switching nodes become unavailable.

As in the VC network, the Internet router must know which node is attached to each physical interface. In the example on slide 60, router 1 is connected to node B on interface 1, node A on interface 2, and router 2 on interface 3. The router must know which interface to use to forward a datagram on the next hop toward the destination. This information is specified in the **routing table**, which specifies an output interface for each range of destination addresses. In the routing table on slide 60, the last line says that a datagram addressed to any destination except A or B should be sent to interface 3 (router 2). Router 3 is called the **default gateway** for router 1.

Slides 61 – 67 Internet Protocol version 4 (IPv4)

The IP frame format is shown on slide 61. The fields from Version to Options should be understood as serial bits (left to right and top to bottom) — they are arranged in a table to fit on the page.

The **Time To Live (TTL)** field allows deletion of zombie datagrams. The source sets the TTL field and each intermediate node lowers the TTL by 1. When the TTL reaches 0, the node must delete the datagram and inform the source.

The **IP address** is 32 bits long — it is well known that the 4 billion possible addresses have been exhausted. The address is written as 4 octets (8-bit numbers in decimal) separated by dots. For example, **`www.hadassah.ac.il = 212.179.79.228`**

The IP address is divided into Network, Subnet, and Host addresses, according to two systems.

Address classes are shown on slide 63.

Class A addresses begin with the octet **0xxxxxxx** (where x = 0 or 1), and this octet gives the network number. The next 3 octets give the host number.

Class B addresses begin with the octet **10xxxxxx**, and two octets give the network number. The next 2 octets give the host number.

Class C addresses begin with the octet **110xxxxx**, and three octets give the network number. The last octet gives the host number. The Hadassah address is class C.

Classless Inter-Domain Routing (CIDR) addresses are written in the form

Octet1.Octet2.Octet3.Octet4/bits_in_network_number

so that in the address **192.168.0.33/27** the first 27 bits are the network address and the remaining 5 bits are the host address. This is shown in slide 64.

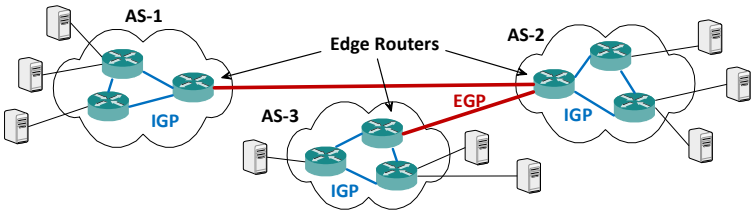
An equivalent system is the **subnet mask**, formed by writing 1 for every bit of the network number and 0 for every bit of the host number. The mask is used to allow hosts to identify other hosts on the same subnet: all hosts on the same subnet will give the same result when a bitwise AND operation is performed between the host address and mask. A host tests the destination address of some other host with the mask. If the remote host is on the same subnet, datagrams are forwarded directly over the local area network (LAN), but if the remote host is on a different subnet, datagrams must be forwarded through a router on the LAN. The division of a local network is shown on slide 66. Host **194.30.5.67** (upper right corner) can send datagrams to host **194.30.5.65** directly on the Ethernet LAN. But to send a datagram to a host outside the local network or to host **194.30.5.129** (lower right corner) it must use the services of the router for subnet 2 (it transfers the datagram to the router on the Ethernet LAN.

Slide 67 shows some reserved IP addresses.

Slide 68 Internet Routing

In order to build routing tables and determine the next hop for an incoming datagram, Internet routers run **routing protocols**. Routing protocols are based on algorithms that treat a set of nodes as a directed or undirected graph.

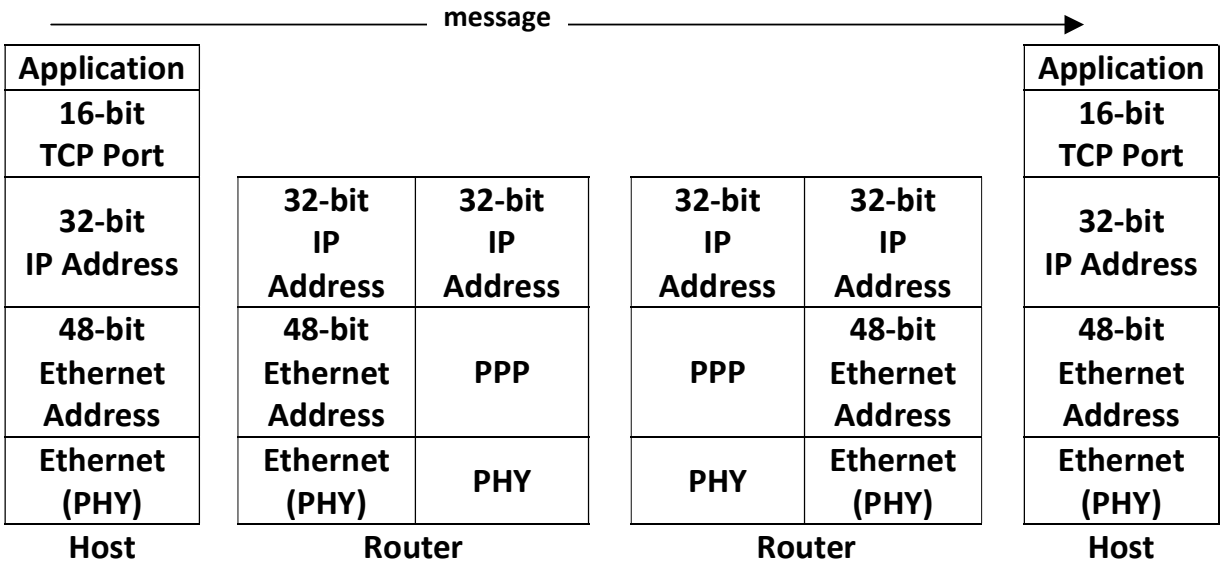
At the highest level, the Internet is modeled as a set of **Autonomous Systems (AS)**. An AS is the set of nodes managed by one organization.



Within any given AS, routing tables in local nodes are determined by one or another **Interior Gateway Protocol (IGP)**, most commonly OSPF. In order to communicate with the global Internet, each AS must have one or more **edge routers (gateway routers)** linked to one or more routers in at least one remote AS. The set of edge routers run a separate **Exterior Gateway Protocols (EGP)** to determine routing tables from AS to AS.

Slide 69 **Relationship of Protocol Layers**

In the most common networking configuration, hosts within an AS are connected by a local area network (LAN), typically Ethernet or WiFi. Host-to-host (application-to-application) messages are addressed using from the source IP addresses to the destination IP addresses. These IP addresses appear in the header of the IP datagram and do not usually change from end-to-end.



In this example, each host is physically connected to a local router (default gateway) through an Ethernet LAN. The sending host forms an IP datagram containing the source and destination IP addresses. The IP datagram is sent to the local router over the LAN using the source and destination MAC addresses defined by Ethernet. Notice that the IP datagram contains the remote host IP address as destination, while the Ethernet frame contains the local router MAC address as destination. This is the reason for **multiple address spaces**.

The sender's local router identifies the next hop router from its routing table and forwards the IP datagram (unchanged, except for the TTL field) to the remote router using PPP or some other appropriate data link protocol (generally **not** Ethernet or WiFi). The remote router receives the IP datagram, identifies the destination host from its routing table and forwards it over the local Ethernet LAN. Notice that the IP datagram contains the sending host IP address as source, while the Ethernet frame contains the local router MAC address as source.

Slides 70 – 73 **Network Address Translation (NAT)**

One remedy for the shortage of IPv4 IP addresses is the use of **Network Address Translation (NAT)**, which permits the reuse of internal (private) network addresses in multiple networks. The address ranges 10.0.0.0 and 192.168.0.0 are defined as private and are not legal assignments for the global IP address of an AS (autonomous system). Therefore, any AS can use this address range internally and use a single global IP address at the output of its edge router. The edge router must receive IP datagrams, exchange IP source / destination addresses in the header, forward the datagram, and maintain a list of translations. An example is shown on slide 70.

No translation is required on traffic internal to the AS. The local address is the real IP address within the AS.

For traffic from a local node to an external node, the outgoing datagram contains:

- **Source endpoint** = local IP address and application source port number
- **Dest endpoint** = remote IP address and well-known port number

The gateway router NAT agent makes two replacements and records the mapping (indexed by the port number assigned by NAT):

- The local IP address is replaced with the gateway's global IP address
- The local application source port is replaced with an unused port number

The gateway forwards the datagram. The remote node replies to the global IP address with the port number assigned by NAT. The gateway router replaces the global IP and the NAT port with the local IP + source port originally chosen by the local application.

An example of the NAT process is given in slide 72.

NAT creates a **traversal problem** — to the outside world, the global IP address of the gateway router is shared by all of the internal nodes. How does a server operating behind a NAT offer service, without a mechanism to address a specific node from the external network? There are three basic solutions:

In **static router referral**, the service port defined on the server is registered in the gateway router. The gateway refers all incoming traffic for this service port to the Server.

Alternatively a **relay server** with a public address (no NAT) is used to coordinate. The client requests service from the server via the relay server. The server must also reply via the relay server. This arrangement is used in services such as Skype. Two clients behind NAT can coordinate a call using the public Skype server.

Finally, **Universal Plug and Play (UPnP)** is a set of specialized protocols for NAT traversal.

Slides 74 – 76 Quality of Service (QoS)

Quality of service (QoS) refers to the level and the type of service offered by a network, as characterized by the following parameters:

Data rate	The physical transmission speed measured in bps
Error control	The bit error rate (BER) over the transmission channel is generally less than 10^{-9} on fiber optic cable, but can be up to 10^{-3} on wireless channels. The packet loss rate depends on the congestion control policy in the intermediate routers (which may randomly discard packets to avoid buffer overflow).
Delay	The end-to-end transmission time for one bit is determined by several factors: • A high data rate produces a lower transmission delay T_{trans} • Congestion / priority control may cause longer queuing delay T_Q • Datagram routing produces a longer processing delay T_{proc} than VC routing.
Delay variation	Dynamic variations ΔT_{trans}, ΔT_Q, ΔT_{proc} in the delay parameters cause jitter that affects streaming service.

Different services require different QoS parameters:

Service	Speed	Error Control	Delay	Delay Variation
e-mail	—	good	—	—
file transfer	reasonable	maximum	reasonable	reasonable
Network File Service (NFS)				
database access				
voice	fast	good	very small	very small
video	very fast	good	very small	very small
real time control				

The speed and delay of **text services** such as email are not relevant — typing will always be slower. Errors are annoying but not critical.

In **remote file access** speed and delay should be reasonable, but there should be no errors.

In **voice, video,** and **real time control**, the speed and delay must be small enough to permit a reasonable user experience, and there should be very few errors. But delay variation is an important parameter in these systems because a sudden jump in delay causes the system to become unsynchronized, leading the system to freeze.

The original Internet protocols (IP and TCP) and the most common LAN protocols (Ethernet and WiFi) offer very limited QoS control. The ATM protocol (see page 19 and slide 53) was designed to address these issues.

Slides 77 – 81 Internet Transport Layer Connections

The Internet transport layer offers two alternative protocols with QoS implications: TCP and UDP.

Transport Control Protocol (TCP) is connection-oriented, establishing a connection before data transfer. It provides error-free delivery — data is delivered in the original order with no errors, duplications, or omissions. TCP provides flow control, varying the sender transmission rate to prevent buffer overflow in receiver, and congestion control, varying the sender transmission rate to prevent buffer overflow in network routers.

User Datagram Protocol (UDP) is connectionless, offering lower overhead and faster service, but with no guarantees of reliable delivery. Segments with errors are discarded with no warning to the application layer.

	Reliable Transport	Unreliable Transport
Protocol	TCP	UDP
Error control	Error-free transmission end-to-end	No error correction
Connection	Connection-oriented	Connectionless
Socket type	Stream	Datagram
Processing overhead	Error correction, packet ordering, congestion control, session state management	None

TCP provides reliable transport and is preferred when its processing overhead makes its use practical. UDP has lighter overhead and provides faster transport with no reliability. Some applications perform light reliability control at the application layer when using UDP. Skype sets up and maintains the communication channel between users over TCP, but transmits digitized voice/video over UDP because lost packets are less irritating than large delay variations (jitter).

TCP and UDP use **port numbers** to identify source and destination applications. When server software comes online, it opens a socket (an operating system service) to allow it to receive service requests. The server agent binds the socket to a **well-known port** between 0 and 1023 that identifies the offered service. A client application seeking service opens a socket (an operating system service) to send requests. The client or the OS binds a port number between 1024 and 65,535 to this socket to identify client application.

For example, a web browser may choose the port number 1025 to locate a web server with the well-known port number 80 (the standard port for HTTP service). The client request is sent from the source port 1025 to the destination port 80, and the server response is sent from the source port 80 to the destination port 1025.

The port numbers permit multiplexing of streams over the same communication channel. For example, the browser may open two channels to the same web server at port 80, one on port 1025 and another on port 1026. The pairs 1025:80 and 1026:80 distinguish the traffic flows.

The frame structure of a **UDP segment** (slide 81) is very simple, containing the source and destination ports, the length of the segment, and a checksum for detecting errors.

Slides 82 – 97 Transport Control Protocol (TCP)

The TCP header (slide 82) contains several fields for reliability control. In addition to the source port and destination port, the important fields are:

Sequence number (SEQ)	The total number of data bytes sent in all previous transmitted segments (not including any data in this segment)
Acknowledgement number (ACK)	The total number of received data bytes received in all previous error-free segments plus 1. This is the number of the next byte expected.
HLEN	The length of the TCP header in 32-bit words
Window size	The number of bytes the receiver can receive now, based on the space in its buffer
Urgent pointer	An offset from SEQ that points to last urgent data byte
Checksum	An error detection code
Options	Options fields and padding bytes to reach an exact multiple of 32 bits

The bits in the flags field are shown on slide 83. The most important are

ACK	Acknowledgment field valid
RST	Reset connection
SYN	Synchronize sequence numbers
FIN	No more data from sender

The flags NS, CWR, and ECE will be described in chapter 5.

The TCP connection set-up (slide 84) is called a **three-way handshake**: the client initiates with a SYN segment, the server replies with a SYN-ACK segment, and the client completes the handshake with an ACK packet. Besides the required flags in each segment, the client and server choose random numbers as an **Initial Sequence Number (ISN)**, the SEQ number for the start of the connection. The random numbers are intended to prevent external hijacking of the connection.

After the set-up procedure, the client may send a segment with SEQ = client ISN + 1, or the server may send a segment with SEQ = server ISN + 1. In each subsequent segment, the sender adds the number of data bytes it already sent to its previous SEQ. In the example on slide 85, the client chose ISN = x and the server chose ISN = y. After set-up the client sends a segment with 500 bytes using SEQ = x + 1 and ACK = y + 1 (the client expects a server segment with SEQ = y + 1). The server receives the 500 bytes of data and replies with SEQ = y + 1 and ACK = x + 1 + 500. The client sends another 400 data bytes with SEQ = x + 501 and ACK = y + 1 and the server replies with SEQ = y + 1 and ACK = SEQ = x + 901.

The sender must hold all sent segments in a buffer until they are ACKed by the receiver. If there is no ACK for some segments then after a timeout interval they are resent from the send buffer. Similarly, the receiver must hold received segments in a buffer until the application layer requests them. Since the buffers are of finite size, the sender and receiver must prevent overflow. The **send window** is the available space in the send buffer and the **receive window** is the available space in the receive buffer. In each TCP segment the sender uses the **window size** field to inform the receiver of the available space in its receive buffer.

The basic TCP sender behavior is as follows:

- The sender starts by setting the values of SEQ (the sequence number for the next segment) and SendBase (the number of the last sent segment that was ACKed) equal to ISN + 1.
- When the sender receives data from the application, it prepares a segment with SEQ.
- If the number of unACKed bytes < SendWindow (the maximum number of bytes that be sent without waiting for an ACK), the sender transmits the segment. If more bytes are "in flight" (sent but not yet ACKed) than permitted, the sender must wait for an ACK before sending.
- The sender updates SEQ = SEQ + length(segment data) to be used in the following segment.
- When the segment is transmitted the sender starts a timeout clock (timer).
- If the sender receives an ACK from the receiver and y is the number of the ACK field, then the sender checks if y > SendBase (does this segment ACK bytes that were not previously ACKed). If so (this is a new ACK), then the sender updates SendBase = y.
- If the number of unACKed bytes is still above 0 then the sender restarts the timeout clock.
- A timeout occurs if the timeout clock reaches zero before the required ACK arrives. The sender then resends all segments starting from SendBase (the byte after last ACKed segment) and restarts the timer.

A simplified sender is described by the following pseudocode:

`// initializations`	
`SEQ = ISN + 1`	
`SendBase = ISN + 1`	`last byte ACKed`
`InFlight = 0`	`bytes already sent but not ACKed`
`RTO`	`timeout interval`
`SendWindow, RecvWindow`	`send & receive windows`
`// main loop`	
`if (new data from application)`	
`  Prepare data segment`	
`    sequence number for segment = SEQ`	
`    SEQ = SEQ + length(data)`	
`  if InFlight < min{SendWindow,RecvWindow)`	
`    Pass segment to IP`	
`    InFlight = InFlight + length(data)`	
`    if !(timer running) timer = RTO`	
`  if (receive ACK with ACK number y)`	
`    stop timer`	
`    if (y > SendBase)`	
`      newACKs = y – SendBase`	`bytes ACKed in this ACK`
`      SendBase = y`	`total ACKed bytes`
`      InFlight = InFlight – newACKs`	`bytes already sent but not ACKed`
`      if (InFlight > 0) timer = RTO`	`reset timeout clock`
`  if (timeout)`	
`    SEQ = SendBase = min{unACKed SEQ}` `    resend`	`set SendBase to last unACKed segment and resend unACKed segments`
`    timer = RTO`	`reset timeout clock`

The basic TCP receiver behavior is as follows:

- The receiver starts by setting expected (the number of the SEQ field expected from the sender) equal to the sender's ISN + 1.
- If the receiver receives an error-free segment containing SEQ < Expected, then the sender must not have received the ACK for that SEQ. The receiver resends a segment with ACK = Expected. The receiver ignores the data in the segment because it was previously ACKed.
- If the receiver receives a segment with error or if SEQ > Expected (an out of order segment), then the receiver resends a segment with ACK = Expected. The sender understands that this ACK is an implied NACK (reject) of the segment and retransmits all segments starting from the expected SEQ. Again, the receiver ignores the segment with the incorrect SEQ.
- If the receiver receives an error-free segment with SEQ = Expected, then the receiver buffers the data for the application and updates Expected to the value it expects to receive in the next segment, Expected + length(segment data).
- The receiver does not send an ACK immediately unless it must (this reduces traffic on the network). The receiver "buffers" the ACK by increasing the value of the ACK to be sent at a later time (the value of the next Expected).
- The receiver sends an ACK if a "delayed ACK timer" expires (usually after 250 msec) or if the number of buffered ACKs is greater than the maximum permitted.
- After an implied NACK is corrected, an ACK is send immediately following a correct SEQ.

A simplified sender is described by the following pseudocode:

`// initializations`	
`Set RecvWindow = receiver buffer size`	
`expected = Sender ISN + 1`	`Expected SEQ from sender`
`ack_buffer = 0`	`received unACKed segments`
`ack_max`	`delayed ACK trigger`
`ack_delay = 250 msec`	`local policy: < 500 msec`
`Start ACK delay timer = ack_delay`	`Delayed ACK: Wait 250 msec for next segment before sending ACK. Send cumulative ACK for a few segments after ACK delay timer counts down.`
`if (ACK delay timer = 0 && ack_buffer > 0)`	
`  Send ACK = expected with updated RecvWindow`	
`  ACK delay timer = ack_delay`	
`  ack_buffer = 0`	
`if (receive segment with SEQ = x)`	
`  if (x = expected && error-free)`	`No errors`
`    expected = expected + length(data)`	`Update expected`
`    if (NACK = 1)`	`This segment recovers from previous error. Send ACK, reset timers and NACK flag.`
`      Send ACK = expected with updated RecvWindow`	
`      ACK delay timer = ack_delay`	
`      ack_buffer = 0`	
`      NACK = 0`	
`    else if (ack_buffer < ack_max)`	`ack_max = maximum number of delayed ACKs. If possible, delay ACK.`
`      nextACK = expected`	
`      ack_buffer++`	
`    else if (ack_buffer = ack_max)`	`If ack_max reached, send cumulative ACK and reset delay timer and ack_buffer.`
`      Send ACK = expected with updated RecvWindow`	
`      ACK delay timer = ack_delay`	
`      ack_buffer = 0`	
`  else`	`On an error`
`    Send ACK = expected with updated RecvWindow`	`Resend the ACK for the last correct segment received, reset ack_delay timer and set NACK flag.`
`    ACK delay timer = ack_delay`	
`    NACK = 1`	

Slide 91 shows a series of data transfers with no errors. The first transfer begins with SEQ = 92 and transfers 8 bytes. The receiver ACKs with 92+8 = 100. The next transfer of 20 bytes has SEQ = 100 and receives ACK = 120. The sender transfers 3 segments with 20 bytes each, with SEQ = 120, SEQ = 140, and SEQ = 160. The receiver sends a **cumulative ACK** with ACK = 180.

In slide 92 one segment arrives with bit errors. The receiver discards the corrupt segment and sends ACK = 160. After a timeout the sender retransmits the segment with SEQ = 160.

In slide 93 the first segment is corrupt and is resent after a timeout.

In slide 94 the first segment arrives correctly but the ACK is corrupt or lost. After a timeout, the sender retransmits the segment. The receiver identifies the segment as a duplicate of the previous segment, discards the duplicate, and resends the ACK.

In slide 95 the sender transmits two segments, but the ACKs arrive late (after a timeout) so the sender retransmits the first of the two segments. The receiver identifies the segment as a duplicate of the previous segment, discards the duplicate, and resends a cumulative ACK. The sender now continues by sending new data segments.

In slide 96 the sender transmits two segments, but the first ACK is lost. The second ACK arrives before a timeout and ACKs all previous data, so the sender continues with new data.

To close a TCP connection, either side may send a FIN segment (segment with the FIN flag set). Slide 97 shows the client sending a FIN segment. The server ACKs the FIN segment, and then sends another cumulative ACK with FIN set. Finally the client ACKs the ACK and the connection is closed.

Slides 98 – 101 Socket Calls — Between App and OS

A socket call requests that the OS allocate a set of data structures to be used for a network connection. The socket connection is between an application program and the OS. Nevertheless, one frequently hears that an application opens a socket to a remote host — this is technically incorrect, but the meaning is clear.

The **socket API** (application programming interface) is a default standard in Linux/Unix but not a formal standard. The basic operations are

Socket	An application requests that the OS allocates resources for a network connection. The OS returns a socket descriptor (socket ID) to the application.
Bind	The server agent or the OS associates the socket descriptor with a port number.
Listen	A server requests that the OS make service available by responding to requests on the bound port.
Connect	A client requests that the OS attempts to connect to a service
Accept	A server requests that the OS accepts a client connect. The OS returns new socket descriptor to the application for that connection. The listening socket continues to listen.
Send	The application sends data to the OS pointed at the socket ID. The OS transmits the data over the associated connection.
Receive	The application requests data from the OS buffer associated with the socket ID. The OS returns data to the application.

In a simple UDP transaction (slide 99),

1. The server opens a socket, binds the socket to a service port, and listens on the socket
2. The client opens a socket, the OS binds the socket to some available port number, the client sends data over the socket, and the OS sends the data the remote endpoint (IP address and service port).
3. Within the sender OS, a UDP agent must accept the data from the application sent over the socket, add the header, send to the IP layer for transmission.
4. Within the receiver OS, a UDP agent must accept the data from the IP layer, perform error checking, and hold the data until the server agent requests it.

The sequence of socket calls from the application layer is shown on slide 100.

A simple server program written in PERL is shown on slide 101.

Slides 102 – 109 Assistive Application Protocols used with the IP Layer

Several application layer protocols are used together with IP:

Domain Name System (DNS) converts between the node name and the numerical IP network address.

Address Resolution Protocol (ARP) is used to locate hardware (MAC) address for a node according to its network address.

Dynamic Host Configuration Protocol (DHCP) is used for IP address allocation on request from a DHCP server, typically located in the gateway router. The server allocates IP addresses from pool of available addresses. The main DHCP message types are:

- DHCP server discovery (host broadcast)
- DHCP offer (server response)
- DHCP request (host request)
- DHCP ACK (server provides address)

An example of DNS is shown in slide 103. Requesting the address for www.hadassah.ac.il, the server provides the machine name (canonical name) and the IP address 212.179.79.228.

DNS is a hierarchical service. A local host requests service from a default name server, which may pass the request to another name server if it does not have the answer.

Some details of the ARP protocol are shown on slide 104. ARP operates in broadcast mode over the local area network within an AS. It cannot provide information about addresses outside the broadcast domain or in another AS. All nodes in the LAN receive the "Who has IP = a.b.c.d" message. Only the host with that IP address answers, including its MAC address.

The **Internet Control Message Protocol (ICMP)** defines messages for management of IP networks. The packet structure is shown on slide 105.

Type 0 messages are used to reply to a request to echo data back to the source — this is used for testing.

Type 3 messages report on various error types — destination network unreachable, destination host unreachable, destination protocol unreachable, destination port unreachable, destination network unknown, or destination host unknown.

Type 8 messages request that data be echoed back for testing.

Type 11 messages report that a datagram has been deleted because the TTL field reached 0.

PING is an application program that tests connectivity to a remote host. The ping client sends echo request packets to a destination IP address (Type = 8 Code = 0). Linux ping sends continuous packets with 56 padding bytes. Windows ping.exe sends 4 packets with 32 padding bytes. To receive an response the remote host must be running a **ping server** (OS service) which responds to each echo request packet with an echo rely packet (Type = 0 Code = 0). The ping client measures the round trip time (RTT) and reports packet losses, RTTs and average RTT. UDP ping sends a UDP request to an undefined and receives an ICMP destination port unreachable message. A ping example is shown in slide 107.

Traceroute is a client application program that sends multiple echo request packets to a destination IP address. The first packet has TTL = 1 and each subsequent packet adds 1 to the previous TTL. Each intermediate router lowers the TTL by 1 and if TTL = 0 it deletes the datagram and sends an ICMP type 11 packet (with router name and IP address) to the sender.

The first traceroute packet has TTL = 1 and is deleted by the first intermediate router. The next traceroute packet has TTL = 2 and is deleted by the second intermediate router. In the end, the traceroute client receives a message from each intermediate node up to the destination host. An example of traceroute is shown in slide 109.

Presentation 2

Access Networking

Access networks connect the user with the global network (Internet). For most users, "the network" refers to the access network and the service they receive from a remote web server or media streaming service. In this chapter we will review some of the technologies that are relevant — but invisible — to users at various layers.

Slide 2 **Small Office / Home Office (SOHO)**

The small office or home office user has a number of devices that connect to each other by a local area network (LAN) that includes a router for connection to the Internet. Slide 2 shows a hub that combines the functions of an Ethernet switch, a WiFi access point, an IP router, and an ADSL modem for connection to Bezeq.

Ethernet (IEEE standard 802.3) and WiFi (IEEE standard 802.11) are LAN protocols defined at the physical and data link layers. ADSL is a physical layer protocol used for fast transmission of bits from point to point, and is used to connect homes and small offices to Internet service providers.

Slides 3 – 12 **Laptop Browser to Web Server**

The simplified view of the client / server connection is shown in slide 3. The user opens a web browser on a laptop to reach a web server. The browser runs an **HTTP client** at the application layer that requests transport service from the OS TCP agent, that requests network service from the OS IP agent, that requests service from the OS WiFi agent, that transmits a request to the router over WiFi. The router (an edge router for the home AS) forwards the IP datagram over the ADSL line to the **Internet Service Provider (ISP)**, which forwards the datagram over the Internet until it reaches the server. At the server, the local router transfers the IP datagram over some protocol at the physical and data link layers to the server. The server data link agent passes the IP datagram to the IP agent, which passes the encapsulated TCP segment to the TCP agent, which passes the HTTP request to the HTTP server agent.

Slide 4 shows some features of the ADSL network. It generally multiplexes high-speed digital data with the usual analog voice channel of a telephone line (at the right side). At the telephone office the voice and data are separated — the Plain Old telephone Service (POTS) is sent to a telephone switch and the data is sent to a DSL Access Multiplexer (DSLAM). The high-speed data is routed to the Internet over the Telco's (telephone company) internal ATM network. Additional details are shown in slide 5.

The technical term for the ADSL modem is ADSL transceiver unit-remote (ATU-R). Slide 6 shows details of an older Bezeq ATU-R, along with translations of some marketing terms to the protocols that implement them.

Slide 7 shows a simplified view of the ADSL implementation by Bezeq. The client is connected by ADSL and ATM to the Bezeq telephone network. Bezeq forwards the datagrams from the client packets to the ISP over an IP network, and the ISP forwards the datagrams to the destination over the Internet backbone.

The make this system work properly a number of intermediate protocols are used. Slides 8 to 11 show the case of a campus in which each user has a separate account with the ISP.

Slide 8 shows the client connection to the router. Between the usual IP layer and Ethernet layers are two additional protocols, PPP for login management and PPPoE to allow each client to establish a private session with the ISP.

Slide 9 shows that the router places the PPPoE frame into an 802.3 Ethernet frame which is then placed inside 3 ATM protocol layers (MPOA, AAL5, and ATM). These ATM packets are sent to Bezeq over the ADSL line at the physical layer.

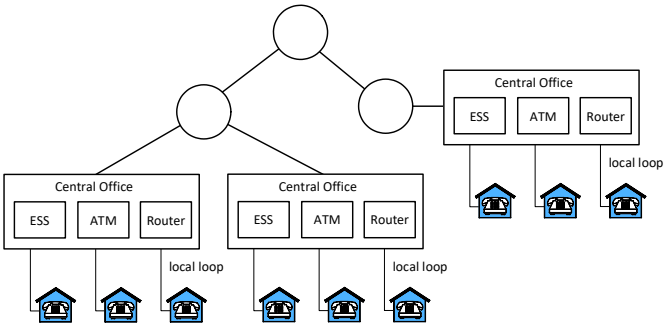
In slide 10, Bezeq forwards the 802.3 Ethernet frame (containing PPPoE and PPP frames) to the ISP over some physical channel. Although these Ethernet frames were only defined for use in a LAN, the use of PPPoE and PPP allows them to be used in the non-local network between Bezeq and the ISP. The PPPoE frames carry PPP packets from the user at home to the ISP over a virtual point-to-point channel — the user conversation with the ISP appears to be direct without intervention by Bezeq. The PPP packets contain NCP packets that allow the user to login to the ISP service with user name and password.

Slide 11 shows that after login, the ISP extracts the user's IP datagrams and forwards them over the Internet in the usual way (between the ISP and server, PPP is just a normal data link protocol).

Slide 12 shows the usual arrangement in which the user's computer runs the standard protocol stack (TCP / IP / WiFi) and the Bezeq router handles the intermediate protocols required to perform transparent IP routing of user datagrams, while maintaining the user connection with the ISP using PPP over PPPoE over 802.3 Ethernet.

Slides 13 – 18 Telephone Network

The largest, oldest and most available wide area network is the telephone network.

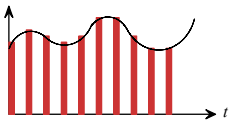


The traditional telephone network is a hierarchy of central offices and higher level switching centers. The **local loop** (so-called "last mile") is the standard analog phone line connecting the Telco central office to nearly every home and office. These lines were designed to carry analog voice and low speed data. The analog voice channel is converted to a 64 kbps bit stream (**DS0**) in the central office.

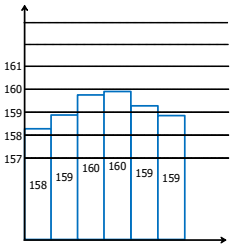
The **PDH / SDH** digital hierarchy combines DS0 digital voice streams for lines operating at data rates of 1.544 Mbps (T1) to 40 Gbps (STM-256). DS0 voice streams are switched between user phones in the **ESS7** switching system, a hierarchical tree of central office and higher-level switches. The more complex (and more expensive) ATM system is a general mesh of packet switches operating up to 2.5 Gbps for combined voice and data streams.

The conversion of **analog voice to DS0 data** streams is performed in three steps:

1. The analog voice is restricted to a bandwidth from 300 Hz to 3300 Hz and sampled every 125 μsec (producing 8000 voice samples per second).



2. The sample stream is rounded off to whole integer values in the range 0 to 255. These values can be represented by an 8-bit number.



3. The bits are transmitted sequentially.
   (8000 samples / second) × (8 bits / sample) = 64 kbps

The DS0 data streams are combined for transmission between central offices in the PDH / SDH **multiplexing hierarchy** shown on slide 15. This means that a single optical fiber running at 155 Mbps can carry 2048 phone calls from Tel Aviv to Haifa at one time.

The **cellular voice network** connects wireless phones to the telephone network using wireless transmission. Geographical areas are divided into **clusters** and clusters are divided into **cells**. In any given cell, a phone connects to a base station by digital radio transmission. The digitized voice (digitized in the phone) is received by the base transceiver, forwarded to a cell controller, then to a cluster controller, and from the cluster to a **mobile switching center (MSC)**. The voice streams are generally sent to the regular Telco network for forwarding to remote locations.

In order to manage **mobility**, the MSC maintains a database (HLR / VLR) that identifies the cluster and cell in which each mobile device is operating. This is required to permit calls to be routed to any given mobile user.

Data traffic is separated from voice traffic at the cluster controller and sent to a Serving GPRS Support Node (**SGSN**), which operates as an MSC for data. The data is forwarded over the General Packet Radio Service (**GPRS**) network, based on IP. The data is sent to a Gateway GPRS Support Node (**GGSN**) and from there to the Internet. The GPRS network manages user mobility for incoming Internet traffic. Slide 17 shows the relationship of controllers in **3G** terminology.

From an economic point of view, the existing telephone network represents an enormous investment in existing equipment, including a global network of hardware nodes and transmission lines developed to provide many services besides Internet. Most services we developed long before the Internet:

- Telegraph — 1794
- Telephone — 1876
- Teletype modem — 1943
- Digital telephone — 1962
- Internet opened to public — 1992

As a practical matter, existing infrastructure is adapted to new uses, and newer models replace older models when older equipment is retired.

Slide 19 Changes in Telephone Business: 2000 — 2008

By the year 2000 it was clear that while business revolves around the data center (data access, storage, processing, and service), employees still conduct business by talking. Mobility had become a standard part of workflow and while video call volume was growing, voice calls still cost money.

Network infrastructure providers (Telcos) saw that their existing infrastructure was designed for voice calls, but their main source of profits was no longer in voice calls, but rather in leasing their telecommunication lines to Internet service providers, private corporate and institutional WANs, and mobile backhaul (the interconnection of cell sites and other cellular infrastructure).

Telco business strategy developed around a plan to scrap the existing PSTN (the hierarchical tree of central offices), sell off the expensive real estate devoted central offices, and use the proceeds to build new a data-oriented mesh network that would support voice calls as just another form of media streaming. This program proceeded smoothly until the worldwide economic crisis of 2008, which began with an international crash of real estate prices.

Slide 20 Next Generation Networks (NGN)

The Telco strategy described in slide 19 was formalized as the **Next Generation Networks (NGN)**, an initiative of the International Telecommunications Union (ITU) for long-term network planning. The **ITU** is an organization within the United Nations that coordinates standards for telephone systems. NGN standardizes the Telco view of technology convergence as **ITU-T Recommendation Y.2001 (12/2004)**:

The NGN is conceived as an all-IP network, in which every service is defined to operate over IP, and all physical infrastructure is defined so to provide efficient support for IP. The NGN is seen as a universal gateway point that dissolves traditional service boundaries (voice versus data, mobile versus fixed, and so on). NGN guarantees universal mobility and promises every service to every user. In order to support different services over IP, extensive new QoS control mechanisms are required, especially IPv6, MPLS, and SIP (service initiation protocol).

Slides 21 – 22 Virtualization in Networking

As seen in slide 17, communication networks were traditionally built from dedicated hardware components, each designed to perform a specific task. For example an IP router was designed and optimized to perform IP forwarding as fast as possible. A virtual system performs the same task as a dedicated system but is implemented in software running on a generic programmable device. Virtual systems must be transparent — it must appear to be a dedicated device to any system to which it provides service and any system from which it requests service.

One example is a virtual LAN (vLAN) in which a programmable LAN switch appears as one or more LAN switches interconnected by a router. The vLAN can segregate LAN segments so that local traffic on one segment does not interfere with local traffic on another, unless the IP addresses require forwarding.

Another example is a Virtual Private Network (VPN) in which a private network is implemented over a public network by isolating private traffic from other users by IP address.

Slide 23 Traditional Networking Functions

Networks can be described as involving two basic functions — forwarding and control.

The forwarding function performs the transfer of data, implementing the network protocols at layers 1, 2, and 3, of the OSI model. In order to achieve highest performance, forwarding is usually performed by dedicated hardware designed specifically for the purpose — Ethernet switches and so on.

The control function involves management of the forwarding function, that is, configuration of the network topology and operating policies, supervision of the running network, measurement, and maintenance. Control is traditionally implemented in the dedicated network hardware by setting mechanical switches, rearranging cable connections, and programming the hardware at a console interface.

Slides 24 — 25 Software Defined Networking (SDN)

A Software Defined Networking (SDN) is a virtual network built from generic programmable hardware. This requires fast implementation of networking protocols and very high I/O capacity. The hardware includes a standardized control interface for configuration by a system programmer, to implement the traditional networking functions — modes, connections, protocols, and policies.

The overall hardware that can be configured to implement an SDN is called the fabric. A networking function implemented in the SDN is known as a logical device.

The data plane (user plane) refers to the aspects of the SDN directly involved in forwarding user data. The control plane handles network configuration data, along with security, maintenance, and other management responsibilities.

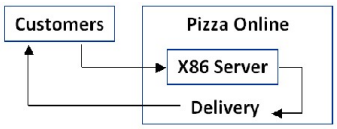
In a Software Define Network, an SDN Controller provides applications with an abstracted view of the logical network assembled as the SDN. Based on this information, SDN application programs communicate their network requirements to the Controller. The Controller translates these requirements to SDN Datapaths — the assembly of physical hardware that implements data forwarding. The Datapath may consist of a single forwarding device, such as a switch or a router, or may be a logical device defined from an internal network of forwarding devices. The hardware assembled to form an SDN is called a fabric.

Slides 26 – 32 Cloud Computing

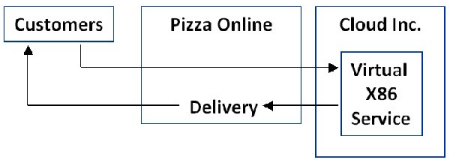
The rapid expansion of **cloud computing** was not anticipated when the NGN framework was being written, but it promotes similar goals. Although cloud computing requires various specialized support technologies, it is more a **business model** than a new form of computing.

The cloud is an outsourcing service model in which a service provider offers computing service to the user as specified in a Service Level Agreement (SLA) that guarantees the service details. In particular the service provider handles the complex tasks associated with **operations, administration, maintenance (OAM)**.

For example, consider a pizza parlor that offers its customers online ordering. The restaurant operator must arrange for a web server, which may require purchase of hardware and hiring of a programmer to maintain the system.



As an alternative, the restaurant operator can arrange the web server to be operated by a cloud service provider who assumes responsibility for the web service in return for monthly payment.



In this sense, the restaurant operator "purchases" computing as a commodity resource just like of water and electricity. The main business advantage is economies of scale, meaning that a large service provider can provide service at a lower cost. Cloud computing also cuts labor and capital costs from the user's balance sheet, which makes investors happy (regardless of the business case).

Cloud service is organized from conventional resources — mainframe computers or server farms with large high I/O capacity running a mix of standard operating systems in native or virtualized environments. The service provider offers a menu of services, a mix of hardware, software, and network capacities. The unique technological issues include:

- Service reliability — the provider is committed to the SLA as a minimum level of service
- The provider must be able to minimize the configuration cost required for the SLA
- The user must be able to identify the SLA requirements to minimize contract costs

Cloud computing is often divided into a hierarchy of service types:

- In **Infrastructure as a service (IaaS)**, the user sees a virtual hardware environment, which may be implemented in real hardware or a virtual machine. The user installs an OS and software to run jobs.
- In **Platform as a service (PaaS)**, the user sees a virtual OS environment, which be implemented as a single OS on a single hardware platform or as a virtual OS. The user installs software to runs jobs.
- In **Software as a service (SaaS)**, the user sees a virtual application software environment consisting of applications running on a private OS or "sandboxed" (isolated) on a shared OS. A sandbox is a private execution environment per application instance.
- In popular **Storage as a service (STaaS)** systems (Dropbox, iCloud, OneDrive, and others), the User sees a virtual mounted storage device. This may be seen as a subset of SaaS.

Among the considerations in cloud computing are cost, reliability, and agility.

In choosing between a setting up a private server or using a cloud service, the user must decide whether operating costs (capital + OAM costs) associated with a private server will be more or less than the operating debt associated with commitment to a cloud contract. The advantages of guaranteed service and simplified growth must be evaluated as a business case.

The provider must find ways to achieve economies of scale to lower the cost per compute job. This can be a difficult task when considering the need for redundant infrastructure for continuity

disaster recovery, and the need to reconfigure service and infrastructure to provide growth, load balancing, and time-zone serving. These needs are simplified through centralized management of OAM, security, and performance. Aspects of centralization are the virtualization that allows multiple users to be served on one physical server, and the multitenancy that provides multiple sandboxed application instances on OS.

In public cloud computing services, the service provider acts as a public utility selling (renting) computing service. The initial service providers, Amazon, Microsoft, Google, and IBM, were leveraging large existing infrastructure, offering a menu of services at fixed prices.

Many organizations have organized their data centers in the form of a private cloud, managed internally or outsourced. The goal is to isolate service developers from implementation issues by establishing a standard development platform. A large corporation can establish a mainframe computer that provides a large number of virtual Windows OS environments for the development of software services offered to the public. In this way, the developers and the public see a standard and familiar software environment, and are isolated from OAM and growth issues in the mainframe infrastructure. This may permit the business to lower costs by maintaining a small group of experts in the infrastructure details and a much larger number of developers who will spend their careers generating new products for the same Windows environment without retraining, promotion or significant salary increases. In Israel, a number of workplace disputes have arisen in connection with this trend.

Cloud computing can be seen as another stage in a **ping-pong** of developments in the computing world over its history:

In the 1950s and 60s, computing was dominated by centralized mainframe computers offering multiple OS instances over a hypervisor. A timesharing OS was used to serve multiple users, so that the user sees a single OS environment via a dumb terminal (now called a thin client).

In the 1970s, user applications were offloaded to a collection of minicomputers offering timesharing services, so that the user would see a timeshared OS environment via dumb terminal.

In the 1980s, user applications were further offloaded to personal workstations (PC) on which the user sees a single-user OS environment running locally.

In the 1990s, it became common (possible) to network single user workstations, so that the user would see a single-user OS environment running locally, but allowing file sharing with other users.

In the early 2000s, centralized control of local the OS environment on PCs was transferred to IT departments, making individual users again dependent on system managers.

In the 2010s, the cloud along with the netbook, tablet, and smart phone have largely reproduced the computing environment of the 1950s, where the old dumb terminal has been replaced by a high-resolution GUI and some limited local computing power.

It remains to be seen how the pendulum swings between centralization and decentralization.

Slide 32 shows a brief history of the changing attitudes toward centralization /decentralization over the past 50 years.

Slide 33 Enterprise Networks

It is unsurprising that large organizations maintain large private networks. These are often called enterprise networks, less because they are managed by large organizations that because the word "enterprise" is intended to suggest grandiosity. In any event, an enterprise network is characterized as:

- Larger, more complex, more expensive, and more "awesome" than a SOHO network
- Combines both LAN and WAN (wide area network) technologies
- Operates at a higher level of network security
- Has a very high overall traffic volume

Slides 34 – 39 Traditional Network Infrastructure

Slide 34 shows an illustration from the 2013 **Cisco Smart Business Architecture (SBA)**, which indicates a number of commercial systems and technologies for organizing a corporate or institutional headquarters connected with branch offices. We will discuss these technologies one by one.

The architecture shows 4 **regions**:

- Headquarters (right side)
- A regional site or main branch (on the top left)
- A remote site or smaller branch or partner (middle left)
- A remote worker

These regions are connected by three types of WAN:

- The Public Switched Telephone Network (PSTN)
- The Internet
- An Multiprotocol Label Switching (MPLS) WAN forming a Virtual Private Network (VPN)

The **headquarters** is further divided into several areas:

- The data center (top left) accessed through main local switches and dial-up PSTN
- Core switches controlling access to the data center
- WAN aggregation (lower left) providing access to the core switches from the Internet and the MPLS VPN
- Internet edge (middle left) providing gateway routers for access to WAN aggregation through a firewall
- Internal users (lower right)

The **data center** contains the main corporate servers, indicated as Cisco Unified Computing System (UCS) rack-mounted servers. This UCS system is advertised by Cisco as "Managing computing, networking, management, virtualization, storage access into a single integrated architecture," providing "a unique architecture that enables end-to-end server visibility, management, and control in both bare metal and virtual environments." On the left is the Wide Area Application Services (WAAS) central manager. Also visible is the Application Control Engine (ACE). These management systems are intended to provide IT managers a view of the

application environment, which is physically implemented across a large number of machines, operating systems, and environments.

The main data switches in the data center are Nexus 5500 — the main switch — and Nexus 2000, which permits expansion of the 5500 switch.

In the WAN aggregation region the MPLS VPN connects directly to the switch because the VPN is inherently secure. The general Internet access is connected through a firewall system or into a Remote Access (RA) VPN, which connects individual users to the data center over the Internet.

Slide 35 shows the conventional model of layered networking for **layer 3 routing** and **layer 2 switching**. Layer 3 (network layer) routing permits universal access to any node by next-hop IP datagram forwarding. Layer 2 (data link) switching is implemented in N x N non-blocking switches and permits access to nodes on the same physical network. Examples of layer 2 switches are:

- Ethernet MAC physical port-to-port switching
- ESS7 64 kbps voice telephone switch
- ATM cell switching for switched and permanent virtual circuits (SVC and PVC)
- Frame Relay switching for permanent virtual circuits (PVC)

Layer 2 switching is faster than layer 3 routing because the routes are set up in advance and there is no need to examine the IP addresses in the datagrams. Large organizations may need to partition their internal networks into **subnets** for administrative and security reasons. In the conventional model, subnets are defined by the IP address mask and routers are required to isolate subnets.

In a virtual LAN (vLAN) the LAN switch is configured in software to partition nodes into subnets with separate broadcast domains. There is no need to insert layer 3 routers to divide the local network into subnets.

In a Virtual Private Network (VPN), encrypted protocols are used to implement a private network on public infrastructure. The VPN creates a "tunnel" between widely separated LAN segments, so that a user in one location can access a node at a remote location as if it were on the same LAN. This is similar to the use of PPPoE in slide 11 to permit a home user to communicate with the ISP over Ethernet as if the user were connected to a LAN at the ISP.

In **Multiprotocol Label Switching (MPLS)** a **label header** is added to each IP datagram. The label can be used to identify a preconfigured route and implement an end-to-end **virtual circuit**. This system requires MPLS-enabled routers, which choose the next hop by the MPLS label (similar to a VC label in a virtual circuit network) instead of examining the IP addresses of each datagram. This saves the time associated with layer 3 processing: accessing the datagram, IP routing, TTL update and so on.

In general, the boundary between switches and routers is becoming less rigid, permitting vLANs and VPNs to be defined quickly in software. An MPLS VPN of the type shown in slide 20 uses MPLS to configure routers to provide private routes — details of the customer's routers are hidden from other customers and the routers in the Internet backbone are not aware of the end-to-end VPN route. Generally, the data traffic is encrypted using IPsec to provide another level of security. **IPsec** is a layer 3 protocol that creates an "encrypted tunnel" in which data and identifying characteristics of the host endpoints are hidden.

Slide 37 shows examples of the core switches shown in slide 34. The **Nexus 7000** switches lines carrying 100 Gbps Ethernet with a total capacity of 15.76 Tbps (15,760 Gbps). The **Nexus 2000** is a fabric extender (FEX) — it can add physical ports to the Nexus 7000 from a remote location.

Slide 38 shows examples of commercial routers. The **XR 12000** can handle 1280 Gbps of incoming traffic, managing all standard IP versions and routing protocols. It can accept infrastructure traffic (physical and data link) from optical fiber (SONET/SDH), Ethernet, ATM, or multiplexed data on copper cable (DS-3/E3). Smaller routers for branch offices are shown below.

Slides 39 – 41 Cisco Software-Defined Access Design

Slide 39 shows a simplified system in the Cisco SDN model. An **underlay network** is the physical hardware used to implement the SDN. An **overlay network** is the logical network model defining the SDN. A database of user accounts (ISE) and hardware descriptions (DNA) provides the system with information about the end uses and capabilities of the network. **Edge nodes** connect user equipment to the SDN and **border nodes** provide the interface with the SDN as an autonomous system.

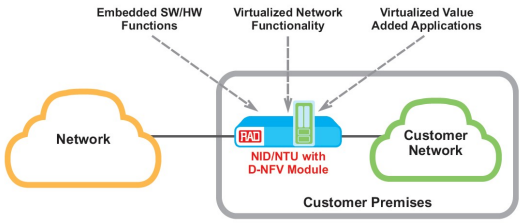
Slide 40 shows a Layer 2 overlay. Here, a physical configuration of (layer 2) switches is used to build an SDN that provides a virtual layer 2 system of a slightly different configuration.

Slide 41 shows a Layer 3 overlay. Based on the same physical configuration of (layer 2) switches on slide 37, this SDN provides a virtual network of switches connected through an IP router that forwards information based in IP addresses within the message section of layer 2 packets.

Slides 42 – 49 What Network Access Providers Do

The activities of network access providers can be seen by looking at the catalog of RAD, a large Israeli company that provides infrastructure for these providers. The key to these activities is management of virtual systems and services. Virtualization permits highly configurable network hardware supporting **Software Defined Network (SDN)** functions.

RAD calls this **Distributed Network Functions Virtualization (D-NFV)**.



The customer network is connected to the WAN (Internet or other) through a **Network Interface Device (NID)** for a general network or a **Network Termination Unit (NTU)** specific to **Carrier Ethernet**. The D-NFV provides various integrated networking and processing functions.

Carrier Ethernet shown in slide 44 is a set of methods for bridging Ethernet LAN segments over a WAN forming a VPN. The combined Ethernet segments, which may be separated over a wide geographical area can operate as a single Ethernet broadcast domain.

We recall that Ethernet (802.3) is a shared medium local area network defined up to about 100 meters with data rates of 10 Mbps to 100 Gbps on copper or optical fiber.

Carrier Ethernet types include:

- E-Line — a point-to-point connection between 2 LAN segments
- E-LAN — a general mesh among several LAN segments

- E-Tree — a hierarchical tree of LAN segments

Security is a major concern for network operators as shown in slide 43. Network access is controlled at the data link layer using the **Network Access Control protocol (IEEE 802.1X)** and **MANsec — MAC layer security standard (IEEE 802.1AE)**. At other layers **Supervisory Control and Data Acquisition (SCADA)** provides firewall protection (FW), intrusion protection (IPS), and anomaly protection.

Slide 46 shows a configuration for connecting a corporate headquarters with remote offices over standard Telco PDH leased lines. Some abbreviations are:

- ETH — Ethernet
- GbE / 10-GbE — Gigabit Ethernet (1 Gbps) / 10 Gigabit Ethernet (10 Gbps)
- DSL — digital subscriber line
- PDH — Plesiochronous Digital Hierarchy (standard telco multiplex hierarchy)
- L2/L3 — layer 2 (data link) and layer 3 (network)
- GPON — Gigabit Passive Optical Network
- ETX — Carrier Ethernet terminal equipment
- POP — point of presence

Slide 47 shows a Carrier Ethernet E-LAN mesh in which the headquarters, remote data center, and branch offices are organized into a single virtual LAN, using Carrier Ethernet over standard Telco PDH leased lines.

Slide 47 shows a configuration for connecting a corporate headquarters with remote offices over a public packet switched network such as IP or MPLS.

Slide 48 shows a configuration for **cellular backhaul** — connecting cell sites with the Mobile Switching Center.

Presentation 3

IPv6

Version 6 of the Internet Protocol (IP) was designed to solve various problems associated with version 4 and provides an essential basis for providing advanced networking services.

Slide 2 IPv4 & IPv6 Header Comparison

Several header fields in IPv4 have been replaced: IHL, type of service, total length, identification, flags, fragment offset, TTL, protocol, and header checksum.

Roughly speaking, type of service is replaced by **Traffic Class,** total length is replaced by **Payload Length**, TTL is replaced by **Hop Limit** (operating in the same way), and protocol is replaced by **Next Header**.

The **Next Header** field indicates the type of header following the IPv6 header — it may be another IP header, used for implementing options, or another protocol.

The **Flow Label** can used as a hint to routers to route packets in the same way.

Slides 3 – 8 IPv6 Addresses

Addresses in IPv6 depend on the scope of their use — link local, site local, or global.

A **link local** address is for use on a local hardware connection over the broadcast domain of a LAN. A **site local** address is for use in the autonomous system (AS) one organization. A **global** address must be legal in the entire Internet.

An address may be defined as **Unicast**, specifying delivery to one address on a single interface, **Multicast**, specifying delivery to a set of interfaces, or **Anycast**, specifying delivery to the closest single interface in set of interfaces.

The IPv6 address is **128 bits**, defining 340,282,366,920,938,463,463,374,607,431,768,211,456 separate addresses (50,000 addresses per square meter of land on Earth). The address is written as a colon-separated **16-bit hex number**, with leading zeros optional (see slide 5) in any 16-bit field. When there are several successive 16-bit 0 fields one (only) continuous field of zeros may be replaced with a double colon (::).

IPv4 compatible addresses are 32-bit IPv4 addresses prefixed with 96 zeros. These are used to identify IPv6 datagrams to be sent into an IPv4 tunnel. An **IPv4 mapped** address is used as the destination address in an IPv6 datagram to be sent to an IPv4 destination. It prefixes the IPv4 address with 0:0:0:0:0:FFFF.

IPv6 address types are determined by the initial bits called a **prefix**. The prefixes 00000000 (binary) and 0000001 are **reserved**. The prefix 11111111 indicates a **multicast address**.

The formats for **link local**, **site local**, and **global** addresses are shown on slide 7.

A link local address is formed by adding a fixed prefix to the hardware MAC address of the node.

A site local address is formed by adding a fixed prefix and a 54-bit administrator assigned subnet address to the MAC address.

A global address is formed by the 3-bit prefix, 45-bit ISP-assigned address, 16-bit locally assigned subnet address, and a 64-bit node address (assigned or derived from MAC address).

The format of a **global unicast address** is

3 bits	45 bits (ISP)	Subnet (16 bits)	64 bits
001	Routing Prefix	Subnet ID	Interface ID (host)

Slide 8 shows an example of IPv6 global addresses:

An **addressing authority** assigns the address range 2001:0410::/32 to the ISP.

3 bits	45 bits (ISP)	Subnet (16 bits)	64 bits
001	Routing Prefix	Subnet ID	Interface ID (host)
001	0 0000 0000 0001 0000 0100 0001 0000		
2001:0410			

The **ISP** assigns 2001:0410:1:/48 to customer 1.

3 bits	45 bits (ISP)	Subnet (16 bits)	64 bits
001	Routing Prefix	Subnet ID	Interface ID (host)
001	0 0000 0000 0001 0000 0100 0001 0000	0000 0000 0000 0001	
2001:0410		1	

Customer 1 can now assign any 64-bit address to each host in its AS.

Slides 9 – 11 Extension Headers

The **next header** field points to the type of header following IPv6 header. Slide 9 shows a list of upper layer protocol headers, such as TCP and UDP, and a list of **IP option headers** for specific services. Some option headers are:

Tunneling	An IPv6 datagram is in data field of this IPv6 datagram. This is used with fragmentation when router adds option field.
Hop-by-hop header	Options that must be checked by every router and destination node
Destination options header	Options that must be checked by the destination node
Routing header	The source sets the route to be taken through the network
Fragment header	Handles fragmentation
Authentication header **Encapsulating Security Payload**	For implementing IPsec

Slide 11 shows examples of chaining IPv6 headers.

Slide 12 **Traffic Class + Flow**

The traffic class field is 8 bits, consisting of 6 bits for the **Differentiated Services Code Point (DSCP)** and 2 bits for **Explicit Congestion Notification (ECN)**.

The DSCP field sets per-hop behavior according to the service required. The ECN field permits routers to indicate congestion to hosts.

A **flow** is a stream of related packets from one source to one destination that requires particular handling by routers, for example real time priority. These requirements can be cached in routers. A 24-bit flow label is chosen randomly from 1 to FFFFF, where 0 indicates that the datagram is not part of a flow. A flow is identified by its label, source IP, and destination IP.

Slides 13 – 17 **Assistive Protocols**

Slide 13 shows the structure and some codes for the version 6 **Internet Control Message Protocol (ICMP)**. New message types include **Router Solicitation** and **Router Advertisement**, **Neighbor Solicitation**, and **Neighbor Advertisement**.

Neighbor solicitation and neighbor advertisement can replace the ARP protocol. To discover the MAC address of a local node, a host sends a neighbor solicitation addressed to the node IP address, with the global prefix replaced with the **link local multicast prefix** FF02:: (64 bits). The neighbor sends ICMP **neighbor advertisement** containing its MAC address as the node ID.

Router solicitation and router advertisement can be used to replace aspects of DHCP. In advertisements, routers send ICMP messages to hosts on some schedule. These are addressed to FF02::1 which addresses all systems on the local link. It provides service configuration parameters, such as maximum transition unit (MTU), the global IP prefix, and the preferred IP address configuration method (ICMP / stateless). In router solicitation, a host sends an ICMP message to request router service addressed to FF02::2, the multicast address for all routers on the local link. Routers respond with a router advertisement.

Slide 16 describes the DHCP operation in IPv6 for host configuration, similar to IPv4 operation.

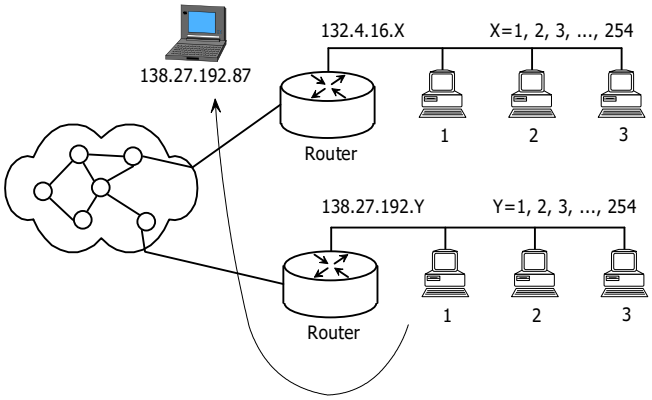
IPv6 also permits **stateless auto-configuration** by which a node can assign itself an IP address that enables communication with a configuration server (see slide 17). The steps are:

1. The node reads its 48-bit MAC address from hardware, splits it into 24-bit slices, and converts it to a 64-bit host ID given by: 24 bits FFEE 24 bits. For example, the MAC address **`11:22:33:44:55:66`** becomes the host ID **`1122 33FF EE44 5566`**
2. The node extends the link local prefix to 64 bits as **`FE80::/10`** $\longrightarrow$ **`FE80::/64`** and prefixes the node address to create a temporary link local unicast address. For example, the host ID becomes the IP address **`FE80::1122:33FF:EE44:5566`**
3. The host verifies that this is a unique address with neighbor solicitation. If there is no response then no other node has this address and it may be used by the host.
4. The host sends a router solicitation. If no routers respond with a router advertisement, the host attempts DHCP. The response message provides network parameters, such as the global prefix for the AS. The host replaces its link local prefix with the global prefix to form a global unicast address.

Slides 18 – 30 Mobile IP

IPv6 permits **management of mobile devices** using fixed IP addresses by implementing Mobile IP, a protocol developed as an optional addition to IPv4. This protocol is a good introduction to mobility management.

The problem to be solved in IPv4 begins with a mobile device, whose home network is a LAN subnet with network number 138.27.192.0. This user requires a fixed IP address to provide accessibility and network services for customers. But the user is mobile and wants to attach to a different LAN subnet with network number 132.4.16.0. The device will not be able to automatically obtain router connectivity, because the router and device are on different subnets with different subnet masks. Also, any datagrams addressed to the device will be routed through the system to the user's home subnet.



The basic idea of mobile IP is similar to call forwarding. The mobile node uses its home router when connected to its home network, but finds a mobile service router when roaming (connected to a non-home network). The home router and mobile service router coordinate between themselves, so that IP datagrams are sent to the usual home router, forwarded by home router to mobile service router, and then forwarded to the mobile node.

A mobile node (**MN**) is any host that can change its point of attachment. Its Home Address is its permanent IP address.

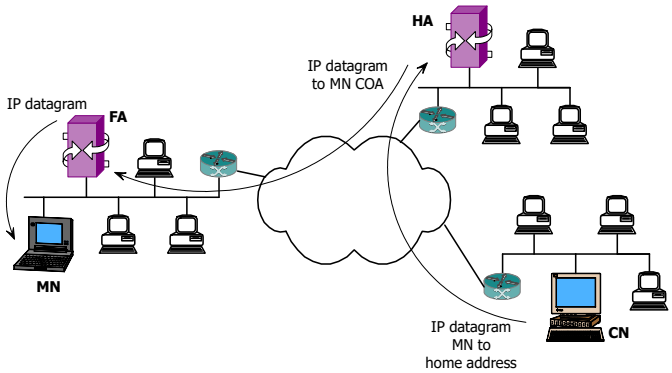
A correspondent node (**CN**) is any node that sends datagrams to the **MN** home address.

The home agent (**HA**) maintains a table of registered mobile nodes and forwards datagrams addressed to mobile nodes.

A foreign agent (**FA**) delivers datagrams between the **MN** and **HA**.

A mobility agent is any **HA** or **FA** supporting mobility.

A Care-of-Address (**COA**) is a FA address used to identify the current location of the MN.



The setup of a mobile connection is as follows:

- To permit discovery, mobility agents broadcast their availability, usually implemented within a router. The mobile node (MN) looks for local router services (HA) when connected to its home network
- When connected to a foreign network, the MN looks for a router advertising Foreign Agent (FA) services.
- The MN requests service from the FA, which assigns a Care-of-Address (COA) to the MN. The FA may have 1 or more available COAs within the IP address space of its subnet, but usually assigns the same COA to all MNs it serves.
- The MN must register to receive service. The FA requests mobile IP support from the MN's home agent (HA). If the HA agrees to provide Mobile IP support, the MN registers its COA with its HA, so that the HA can forward datagrams to the FA using the COA.
- Datagrams from a correspondent node (CN) arrive at the HA by normal routing (they are addressed to the MN).
- The HA encapsulates the arriving datagram in a tunnel datagram by adding a new IP header (the original datagram is treated by intermediate routers as meaningless data). The new IP datagram is addressed to the COA address from the HA address and arrives at the FA by normal routing.
- The FA recognizes the COA and removes the original datagram from the encapsulated datagram. This datagram must be handled carefully — it cannot be routed by the normal procedures of the routing table because the destination address is on the home network of the roaming MN. The FA places the datagram in a LAN data link (layer 2) frame addressed to the MN on the FA's network (even though the destination address in the MAC frame and the destination address in the IP datagram disagree).

Another complication is further roaming and **hand-off**. Suppose the roaming MN moves to new attachment point (network), so that it requires a change of FA. In this case, to avoid datagram loss (and to avoid very frequent registrations) the MN registers and maintains multiple bindings with multiple COAs, the old and new. The HA can now forward each packet to multiple COAs and the MN receives packet at one COA. The process is shown on slide 27. Route optimization is possible after a reconnection.

Triangle routing is the condition in which the CN sends datagrams to the HA, which forwards them to the FA for the MN, but the MN sends datagrams directly to the CN. **Route optimization** is possible if the CN is enabled to run mobile IP. In this case,

1. The CN sends data to the MN in its home network.
2. The HA advises the CN that the MN is mobile.
3. The CN requests a binding so that it can send datagrams directly to the MN at the COA.
4. The HA provides the COA to the CN.
5. The CN sends datagrams to the MN directly using the COA.

Slide 30 shows the process of roaming under route optimization.

Slides 31 – 32 IPv6 Mobile IP

With the growth in wireless mobile networking, mobile IP became a priority for IPv6.

Using the addressing features of IPv6, the mobile node (MN) can always form a **link local unicast address** from its MAC address and a **roaming address** (from the global prefix and MAC address) using the stateless auto-configuration procedure. No special Foreign Agent (FA) is required because the MN uses its roaming address as a care-of-address (COA).

The MN registers with its Home Agent (HA) by sending a binding update containing its roaming address. The HA forwards traffic for the registered MN by tunneling packets from the CN to the MN. The MN sends packets to CN directly and route optimization is achieved by the HA providing the CN with the COA.

IPv6 provides further support for mobility using option headers:

- Any node can write the home address for a MN in the **destination option header**. By examining the option header, the destination MN can identify datagrams by its home address.
- Tunneling can be accomplished by using the IPv6 routing extension headers instead of encapsulation, which reduces processing time.
- When an MN moves from one ESS to another in a WiFi network, there is a layer 2 handover (a change in the attached access point and associated ESS ID). A router node can detect this change in the subnet prefix and automatically updates the CoA.
- IPv6 Mobility header messages are provided for Home Test Init, Home Test, Care-of Test Init, and Care-of Test, binding update and acknowledgement.

Presentation 4

Basics of Wireless Networking

For over 25 years wireless networks are part of our day-to-day experience and the basis for mobile communication that has changed many patterns of behavior. Despite the simplicity of using mobile telephones and tablets, the technology behind mobile communication is more complex by several orders of magnitude. This chapter presents a simplified introduction to radio communication and the two principle forms of wireless mobile networking — Wi-Fi and cellular. We will pay particular attention to the common features of the two contexts — the local and global networks.

Slide 2 **Energy and Power**

Two essential concepts for understanding radio transmission are energy and power. **Energy** is defined as the ability to do work, and may be kinetic (energy of motion) or potential (stored energy). **Power** is the quantity of energy used or transferred per second. Power is measured in units of **Watts** and energy is measured in **Joules**, defined as Watts × seconds. Power purchased from the electric company is paid for in units of kilowatt-hours, where

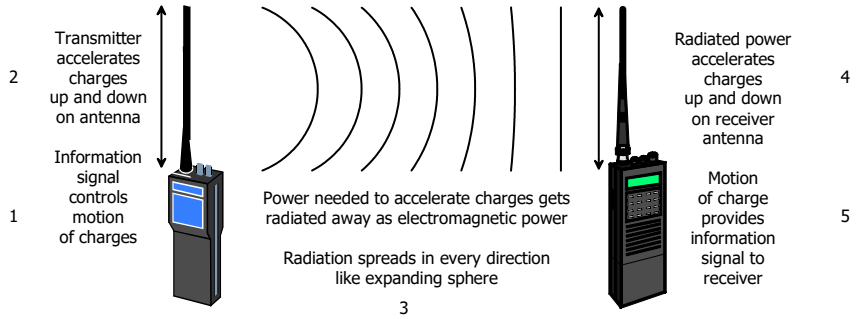
1 kW-hour = 1000 Watts × 3600 seconds/hour
= 3.6×10^6 Joules

Slides 3 – 4 **Electricity and Magnetism**

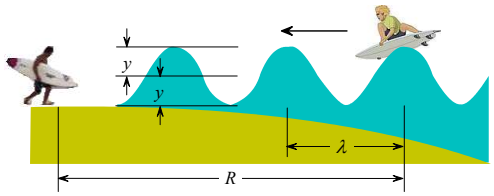
A charged object, such as an electron in motion may create an electric field **E** and a magnetic field **B**. These fields are vector quantities in 3 dimensions, and their magnitude and direction generally depend on the time t and location **x** at which they are measured. **Radiation** is the transfer of power from a charged object through its electromagnetic field. This radiation can perform work when it is absorbed by another object, the process underlying radio transmission. The radiation field associated with an electromagnetic field is proportional to the vector product **E** × **B** and this is the power emitted by a moving charged object.

A motionless charge produces an electric field with magnitude $E = -\frac{\text{charge}}{R^2}$, where R is the distance of the charge at the point of measurement, and magnetic field B = 0. Since **E** × **B** = **0** this system does not radiate.

In a radio, charges are accelerated back and forth along the length of the antenna. Electric charge in motion is called an **electric current**, and in an antenna the charges accelerate to produce the time-dependent current. Both **E** and **B** are non-zero, producing a radiation field that can be detected in a remote antenna.



Electromagnetic radiation satisfies a wave equation, whose solution shows that the radiated power depends on time t and distance R from antenna. Electromagnetic waves can be compared to **ocean waves** rolling onto a beach.



The wave height has peaks and troughs, and we take y to be the height of peak above the average water height, which is also the depth of trough below center.

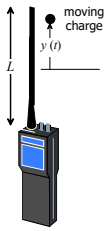
At a fixed distance from the shore, wave rises and falls over time — the period T is time between the arrivals of two wave peaks. The frequency f of wave arrivals is given by 1/T, the number of wave peaks per second.

At some fixed time, multiple wave peaks are present at various distances from the shore. The wavelength λ is the distance between two wave peaks.

A **surfer** can ride the peak of a wave, because the peak depends on the distance from the shore and the time. The speed of the moving peak is given by f × λ.

By analogy, a **charge on a radio antenna** accelerated up and down oscillates from top to bottom over a distance L every T seconds.

The frequency of the oscillation is f = 1/T so that the position y(t) of the charge at time t can be written



$$y(t) = \frac{L}{2}\cos\left(2\pi\frac{t}{T}\right) = \frac{L}{2}\cos(2\pi ft)$$

The properties of electromagnetic field are given by Maxwell's equations, which leads to a wave equation. The solution to the electromagnetic field of a straight antenna is

$$\mathbf{E}(R,t) = \frac{\mathbf{E}_0\cos\left[2\pi f\left(R/c - t\right)\right]}{R} \qquad \mathbf{B}(R,t) = \frac{\mathbf{B}_0\cos\left[2\pi f\left(R/c - t\right)\right]}{R}$$

R = distance from antenna to point of measurement

t = time (measured on some clock)

f = frequency $= 1/T$

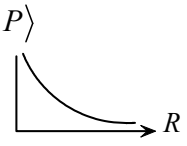
$\mathbf{E}_0, \mathbf{B}_0$ are physical constants

c = speed of light

The radiated power vector and the average power (averaged over one period) are

$$\mathbf{P} = \mathbf{E}\times\mathbf{B} = \frac{\mathbf{E}_0\times\mathbf{B}_0}{R^2}\cos^2\left[2\pi f\left(R/c - t\right)\right] \qquad \langle P\rangle = \frac{1}{T}\int_0^T |P(t)|\,dt = \frac{|\mathbf{E}_0\times\mathbf{B}_0|}{2}\frac{1}{R^2}$$

Because the power is proportional to $1/R^2$, the signal strength **fades** — it decreases with distance from the source antenna.



The electromagnetic wavelength is defined as $\lambda = cT = c / f$, so that $\lambda f = c$, the speed of light. Using this definition, the electric field can be written

$$\mathbf{E}(R,t) = \frac{\mathbf{E}_0 \cos\left[2\pi f\left(R/c - t\right)\right]}{R} = \frac{\mathbf{E}_0 \cos\left[2\pi\left(\frac{fR}{c} - ft\right)\right]}{R} = \frac{\mathbf{E}_0 \cos\left[2\pi\left(\frac{R}{\lambda} - \frac{t}{T}\right)\right]}{R}$$

and the wave peaks are found when $\cos\left[2\pi\left(\frac{R}{\lambda} - \frac{t}{T}\right)\right] = 1$, which is satisfied when

$\left(\frac{R}{\lambda} - \frac{t}{T}\right) = 0, 1, 2, \ldots$

Therefore shifting in time and distance from one wave peak to the next requires $\left(\frac{\Delta R}{\lambda} - \frac{\Delta t}{T}\right) = 0$.

The speed of the wave is the change in distance divided by the change in time from one peak to the next, so that $v = \frac{\Delta R}{\Delta t} = \frac{\lambda}{T} = \lambda f = c$, which shows that the **wave peaks move at the speed of light**.

Looking at the electric field at some fixed distance R_{const} it takes the form

$$\mathbf{E}(R_{const}, t) = \frac{\mathbf{E}_0}{R_{const}} \cos\left[2\pi\left(\frac{R_{const}}{\lambda} - \frac{t}{T}\right)\right] = C_1 \times \cos\left(2\pi ft - C_2\right)$$

where C_1 and C_2 are constants, which rises and falls over time.

Looking at the electric field at some fixed time t_{const} it takes the form

$$\mathbf{E}(R, t_{const}) = \frac{\mathbf{E}_0}{R} \cos\left[2\pi\left(\frac{R}{\lambda} - \frac{t_{const}}{T}\right)\right] = \frac{C_1}{R} \cos\left[2\pi\left(\frac{R}{\lambda} - C_2\right)\right]$$

where C_1 and C_2 are constants, which rises and falls, but also fades over distance R.

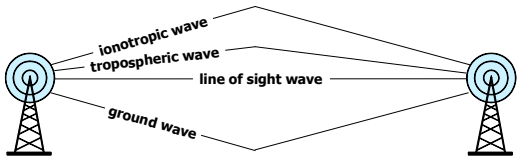
Radio antennas are most effective in the frequency range from 30 kHz (wavelength λ = 10 km) to 300 GHz (wavelength λ = 1 mm). As shown on slide 10, chemical reactions generate higher frequencies: infra-red (IR), visible light, ultra-violet (UV), X-rays (Roentgen) and nuclear reactions generate gamma rays.

A line antenna is most efficient when $L = \lambda / 2$. GSM phones operate at a frequency of about 1 GHz, which is equivalent to a wavelength of $(3\times10^{10}$ cm/sec)/(10^9 Hz) = 30 cm. This is convenient for a phone about 15 cm in size.

Slides 12 – 17 Radio Wave Propagation

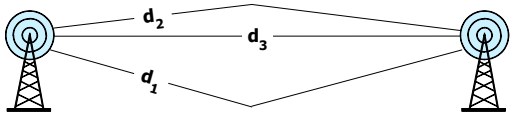
A radio transmitter generates radio waves, which propagate (spread out) through space. Part of the radiated power is generally obstructed or reflected, and another part is detected by receiver.

The radio transmission that propagates directly to the receiver is called the **line of sight wave**. The **ground wave** is a radio signal that reflects and propagates along the ground. Other portions of the transmission **refract** off different layers of the atmosphere, the lower troposphere and higher ionosphere.



The longer the propagation path, the longer the delay of the arriving signal.

In addition to refraction in the atmosphere (bounding off the sky), the propagated wave may be reflected back to the transmitter (as with a mirror) or absorbed in trees, walls, and buildings. Absorption weakens the line of sight wave, making it less effective. Refraction causes the transmitted signal to be received in multiple copies separated by various time delays, because a longer transmission path leads to a longer transmission time. These multiple delay times, compared to the arrival of the line of sight wave, cause a phenomenon called **multipath fading**.



At any time t, the signals arriving at the antenna on the right have traveled over three paths of lengths $d_1 > d_2 > d_3$. The **line of sight wave** travels over the shortest distance d_3 and was the last to be transmitted. The refracted **atmospheric wave** traveled the longer distance d_2 and so it must have been transmitted earlier to arrive together with the line of sight wave. The **ground wave** travels the even longer distance d_3 and so it must have been transmitted even earlier than the atmospheric wave to arrive at the same time as the other two. Since the atmospheric wave and ground wave were transmitted at different times than the line of sight wave, they contain different information and can interfere with the main information stream.

Interference can also cancel the line of sight wave. Slide 15 shows how two waves on a string can propagate in opposite directions and be detected at opposite ends of the string. But in the very center of the string, the two pulses exactly cancel each other — at that precise point no pulses will be detected. This is an example of how moving a transmitter or receiver slightly can change reception quality.

A more detailed demonstration is given on slides 16 and 17. Two waves arrive at a receiving antenna from two slightly separated sources (R,t) and (R+ΔR,t+Δt). Making various approximations and using trigonometric identities, the electric field becomes

$$\mathbf{E} = \frac{\mathbf{E}_0}{R}\left\{2\cos\left[2\pi\left(\frac{R}{\lambda} - ft\right)\right] \times \cos\left[\pi\left(\frac{\Delta R}{\lambda} - f\Delta t\right)\right]\right\}$$

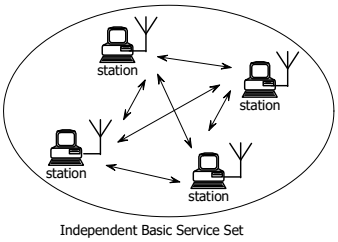
which has the form of a single-source electric field multiplied by a factor depending on the separation of the two sources. In a completely transparent medium (such as empty space) the argument of the second cosine is 0 and the cosine is 1. But under various circumstances it can occur that the cosine is zero and the electric field of the transmission vanishes. This can be a serious problem in WiFi, and for this reason most modern access points have a double or triple antenna that can reduce the multipath fading.

Slides 19 – 22 IEEE 802.11 Protocol Layers

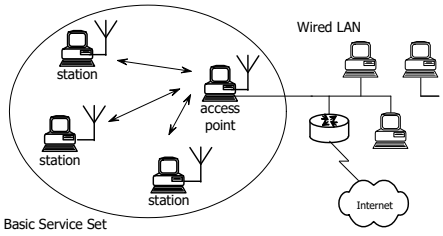
IEEE 802.11 is a large collection of standards for wireless local area networks. The main standard is defined at the physical and data link layers. Within the data link layer, the MAC sublayer defines required and optional features of a wireless LAN, and IEEE 802.2 is used for optional logical link control (common to Ethernet and other 802 standards).

WiFi is a trademark of Wi-Fi Alliance trade association and describes a collection of specifications for mass market, consumer wireless LANs based on 802.11 standards. WiFi can be configured in 3 basic operating modes.

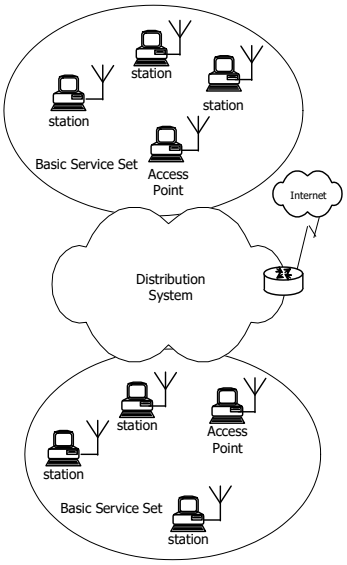
In WiFi **ad hoc mode**, an Independent Basic Service Set (IBSS) is any set of 802.11 STAs (wireless stations) that transmit and receive on the same frequency. This mode provides peer-to-peer service with no access point and no connection to an external network. The STAs communicate directly with one another, a useful service for quick set up, such as emergency response in a remote location. No authentication or registration is required to join the IBSS. Multiple IBSSs are completely independent, with no bridging or hand-off between them.



In WiFi **infrastructure mode**, a Basic Service Set (BSS) is a set of wireless end stations (STA) and an Access Point (AP) that is connected to external (usually wired) network infrastructure. The AP acts as a base station for the wireless network — all traffic flows through the AP with access permission decided by contention (competition) or polling (the AP grants access by some schedule). To join the BSS, stations must first associate with the AP through user authentication and registration. This is the most common configuration for home WiFi systems.



In WiFi **extended infrastructure mode**, an Extended Service Set (ESS) is two or more BSSs interconnected to form a single subnetwork (broadcast domain) that looks like one large BSS to the LLC sublayer. There is one Access Point (AP) in each BSS, and the BSSs are connected with one another via a Distribution System (DS). The DS is the backbone network, performing MAC-level transport of MAC SDUs, but its implementation details are left general (they are not specified in 802.11). The **portal** is a software gateway function in the AP that bridges the BSS to any non-802.11 DS protocol. The DS services permit handoff (a station moving from one BSS to another) by providing coordination between APs.



Slides 23 – 27 WiFi Medium Access

Medium access in WiFi is affected by connectivity problems not seen in cable systems.

The **hidden node problem** describes four wireless stations A, B, C, D in which A begins transmitting to B. C is too far from A to notice its transmission and starts transmitting to D. But C is in range of B so that the C to D transmission interferes with the A to B transmission.

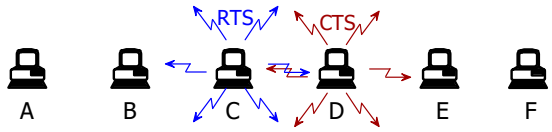
The **exposed node problem** describes four wireless stations A, B, C, D in which B begins transmitting to A. C is in range of B, notices its transmission, and waits before transmitting to D. But D and B are out of range and A and C are out of range, so this wait is inefficient.

WiFi specified two medium access methods.

In **CSMA with Collision Avoidance (CSMA/CA)**, stations listen for transmissions and do not transmit if any carrier signal is detected. This method is exposed to the hidden node problem because no collision detection is possible (the antenna cannot receive while the transmitter active). To avoid collisions, the stations practice non-persistent access with random backoff, which means that when they see an opportunity to transmit they wait a random time interval before beginning.

Multiple Access with Collision Avoidance (MACA) is a connection-oriented technique with additional overhead but better throughput in hidden and exposed nodes. Before data transmission begins a channel is set-up. The source station initiates a Request To Send (RTS) message containing a Net Allocation Vector (NAV) that specifies the predicted data transmission time. The destination station responds with a Clear To Send (CTS) message that echoes the NAV. Following error-free transmission of the data, the destination sends an ACK message.

With 6 stations



C transmits to D — first C sends RTS and then D returns CTS. At the same time B has data for A and E has data for F. The **exposed node problem** (B to A) is solved: B hears the RTS but not the CTS, so it knows that C's destination is out of range (and no interference will occur if it transmits). The **hidden node problem** (E to F) is solved: F hears the CTS but not the RTS, so it knows that the sender to D is out of range but D is in range, so it must wait before transmitting.

Slides 28 – 37 MAC Sublayer Frame Structure

The MAC layer frame in WiFi consists of the header, data, and frame check sequence (FCS).

Frame Control	Duration/ ID	Address 1	Address 2	Address 3	Sequence Control	Address 4	Frame Body	FCS
2 bytes	2 bytes	6 bytes	6 bytes	6 bytes	2 bytes	6 bytes	0-2312 bytes	4 bytes

The header fields are:

Frame Control	Control flags (see slide 29)
Duration/ID	Timing control (NAV)
Addresses	Various MAC entities whose address is required for end-to-end transmission
Sequence Control	Sequence/Fragment number for error/flow control

The MAC header contains up to 4 address fields describing up to 5 MAC entities:

1. BSS Identification Number (BSSID) MAC address of AP (or random in IBSS)
2. Source Address (SA) Station that initiated the message
3. Destination Address (DA) Final destination for the message
4. Transmitting Station Address (TA) Source station for the message on this hop
5. Receiving Station Address (RA) Destination station for the message on this hop

These MAC entity addresses are used in address fields 1 to 4 depending on the transmission scenario (to DS and from DS are flags in the frame control field of the MAC header):

	To DS	From DS	Address 1	Address 2	Address 3	Address 4
No AP in use	0	0	DA	SA	BSSID	—
From STA to AP	0	1	DA	BSSID	SA	—
From AP to STA	1	0	BSSID	SA	DA	—
From AP to AP	1	1	RA	TA	DA	SA

In all cases, the address fields refer to:

- Address 1 Immediate destination address
- Address 2 Immediate source address
- Address 3 Final destination or source when DS performs distribution
- Address 4 Source address for DS to DS messages (802.11 is also DS)

Slide 32 shows address in an IBSS with no AP in use. The to/from DS bits are 0, the source and destination addresses are stations, and the BSSID is included for error protection.

Slide 33 shows addressing in a BSS. When the AP sends data to a station, the immediate destination is the final destination, the immediate source is the AP (with MAC address BSSID) and the original source was the station with address SA. When the AP receives data from a station, the immediate destination is the AP (with MAC address BSSID), the immediate source is the original source with address SA, and the final destination is the station with address DA. Slide 34 shows an example of a frame sent from station A to station B via the AP in two hops.

Slide 35 shows control and management frames in a BSS. These are sent with both DS flags set to 0, but the AP is either source or destination (and included in address 3 for error protection).

Slides 36 and 37 show addressing in an ESS. As in a BSS, all transmissions are sent through an AP, and the same address fields are used in the same way. On the first hop, the addressing is "To AP 1 from Station A for Station B" and on the last hop, the addressing is "To Station B from AP 2 from Station A". WiFi does not describe how the frames are transmitted from AP 1 to AP 2.

Slides 38 – 39 Mobility Management

As seen in Mobile IP (slides 18 – 30 of presentation 3) **mobility management** has certain features common to all infrastructures:

Registration and **authentication** of user credentials is performed when a mobile station (MS) is activated in a given service area. In WiFi, the access point (AP) manages registration by requiring a password. In the cellular system (GSM / 3G / 4G), databases are maintained on individual users. The Home Location Register (**HLR**) maintains account and location information for home customers (users with a contractual relationship to the server provider). A Visitor Location Register (**VLR**) is a cache of HLR data on active roamers in any particular service area, allowing the system to locate the user at any given attachment point.

Call establishment is performed when a user initiates or receives a call in a service area (in the home service area or as a roamer).

Security (encryption) is intended to protects users from fraud and eavesdropping.

Handoff (handover) is performed when the MS changes its attachment point to the network. In WiFi, this means switching AP. In cellular, this is the complex process of moving from cell to cell.

In either case the process can be performed in one of two ways. In **hard handoff**, the old cell transfers control to the new cell in a break-before-make sequence. The transceiver (transmitter/receiver) in the old cell stops transmitting to the user and then the transceiver in new cell begins transmitting to the user. In older systems this cause many calls to drop. In **soft handoff**, the central transceiver (primary connected base station) coordinates with the nearest APs or cells and determines which transmitter is receiving the strongest signal from the user. In a make-before-break sequence, the transceiver in the old cell continues transmitting to the user while the transceiver in new cell begins transmitting to user. When the transceiver in the new cell is safely connected user, the old cell stops transmitting. Newer systems use soft handoff to prevent call drops and other inefficiencies.

Slide 41 1970 — 0G Mobile Phone System (MPS)

The first mobile phone system (zeroth generation) deployed one central transceiver (base station) and mobile telephones that could communicate by radio. The system transmitted at high power for maximum distance, covering 65 to 80 km.

The radio modulation scheme was standard analog FM, supporting 12 simultaneous mobile phone calls. If all 12 channels were busy, other calls were blocked. This required 24 carrier frequencies, with 2 frequencies per phone: one dedicated transmit frequency and a separate dedicated receive frequency.

Slides 42 – 46 Cellular Concept

The essential idea in cellular telephony is to divide the coverage area into small regions called cells. In each cell, a central cell transceiver serves all users in the cell. All mobile stations communicate via the base cell transceiver, so that each active device must be allocated a frequency pair: a **downstream** frequency to receive from the base station and an **upstream** frequency for transmitting to the base station.

The cellular system can transmit at relatively low power, just enough power to cover one cell. Therefore the system can **reuse** the same set of frequencies in many cells with no interference between cells.

Slide 43 shows reuse patterns in cellular systems. In the 7 cell reuse pattern, cells are designated A, B, C, ... , G. The full set of radio channels available for the system are divided into 7 sets and each cell type uses only frequencies of its type. The pattern guarantees that no A cell borders on another A cell, which increases the distance (and reduces the interference) between A cells (using the channels allocated to cells designated A).

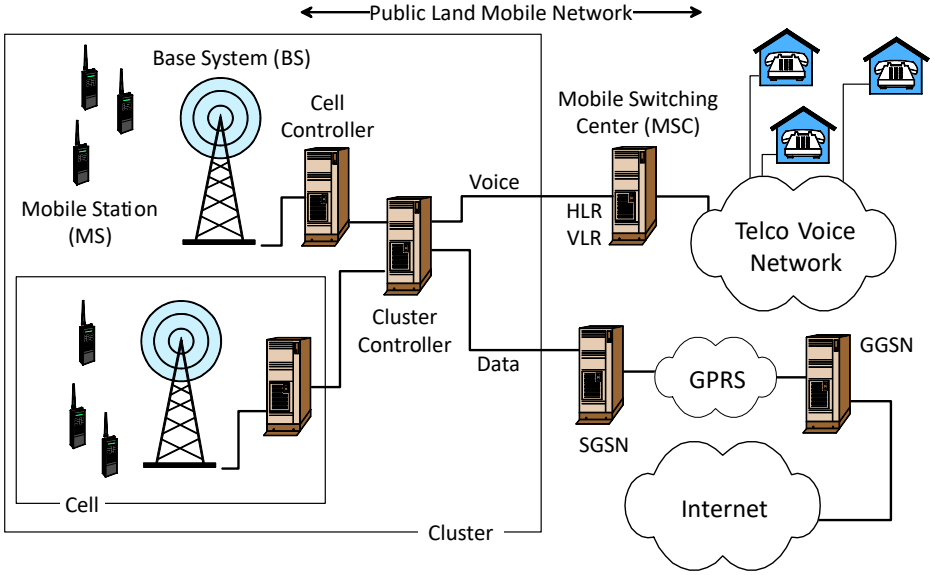
As shown on slide 44, a mobile service provider operates in some set of service areas (in some parts of the world — including the USA — large geographical areas have no cellular coverage). A service area is divided into clusters — each cluster is based on the reuse pattern, so that all available channels are used in each cluster. The cluster is divided into cells.

The generations of cellular networks are shown in slide 45.

- The 1G networks were proprietary systems operated by service providers. These systems used analog radio transmission and could not operate with other systems.
- The 2G networks in the US were upgrades of proprietary 1G systems, providing service to more uses but still no interoperability.

- The 2G systems in Europe were based on standards designed to permit a user to use the same device anywhere in the EU. The GSM system became the basis for most cellular network architectures since the 1990s, using time division multiplex to combine 8 digitized voice channels per frequency band on a frequency division multiplex of the analog radio capacity. CDMA is a more efficient multiplex system in which each user is assigned a different encoding scheme. Both GSM and CDMA provided voice service over traditional circuit switched digital channels. Separate from the voice channel is the GPRS packet switched network that enabled IP-based data to be exchanged with the Internet. In 2G systems, each individual cell was connected to a centralized switching center that connects the user device to the non-mobile telephone network and controls handover.
- Following the success of GSM, 3G systems in most of the world implemented UMTS (a GSM enhancement) or WCDMA (a CDMA enhancement). In addition to higher transmission speeds and higher reliability (fewer dropped calls), 3G systems improved handover when a mobile device moved from one cell to another. Unlike 2G, individual cells were organized into lower level groups that could provide handover without intervention by the centralized switching center.
- 4G systems were defined to satisfy Next Generation Network requirements. LTE and WiMAX are called 4G systems, and although they provide higher speeds, they do not meet the NGN requirements. In 4G networks, the original circuit switched network for voice calls is eliminated. Voice calls are transmitted in packet mode as Voice over IP (VoIP).
- 5G is defined in two stages. The initial implementation is a radio enhancement to 4G LTE. The full design provides for highly abstract Software Defined Networks (SDN).

Slide 46 shows the general architecture of a 2G cellular network, which became the basis for understanding the network functions is all subsequent generations.



- A cell (base system) consists of a radio antenna and a controller called a Base Transmitter System (BTS).
- The BTS of each cell connects to a cluster controller called a Base Controller System (BSC).
- The BTS separates digitized voice traffic from data traffic.

- Voice calls are sent on circuit mode channels to a Mobile Switching Center (MSC) for connection to the PSTN, which interconnects cellular service areas.
- Data is sent to a Serving GPRS Support Node (SGSN), a packet equivalent of the MSC, for connection to a Gateway GPRS Support Node (GGSN) for connection to the general Internet.

Slides 47 – 50 GSM Registration Process

The registration process is similar in GSM and UTMS (3G):

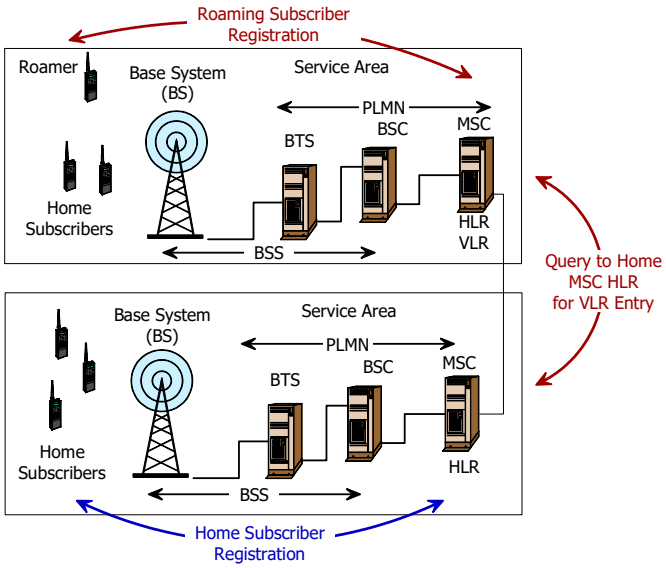
- An MS enters a service area and establishes a low bit-rate control channel with the service provider.
- The MS requests service and the BTS allocates a frequency pair (upstream and downstream) for the registration process.
- The MS reports its location, status, and identity to the Mobile Switching Center (MSC). The user ID consists of a dedicated hardware ID code in the phone and an ID from the **Subscriber Identity Module (SIM)** card identifying the customer. The MS generates an encrypted access code for the network, called a Temporary Mobile Subscriber Identity (TMSI).
- The Mobile Switching Center (MSC) authenticates the customer identity with the HLR (or for roaming subscriber, creates a VLR entry), updates the HLR and billing database.
- The MS releases the frequency pair and maintains contact with the BTS over the low bit-rate control channel.

The registration process is summarized in slide 48.

The process of **call establishment** is shown in slide 49 (for a call initiated by the mobile station) and slide 50 (for a call incoming to the mobile station).

Slide 51 Mobility Elements in GSM

Mobility management in GSM conforms to the pattern shown below.



The user operating in the **home service area** uses a control channel to register with a base station in a cell. The cell consists of a base transceiver (BS), a Base Transmitter System (BTS), the controller for the cell. The cell BTS forwards the registration request to the cluster controller called the Base Controller System (BSC). Together, the BTS (cell) and BSC (cluster) are called the Base Station Subsystem (BSS). The cluster forwards the registration request to the Mobile Switching Center (MSC), which contains the HLR. The MSC accepts the request, registering the user in the cell.

If the **user roams** to the service area of another provider, the user must submit a new registration request. This time, the request is passed to the MSC and from there to the MSC of the home service provider. If the home service provider permits roaming for the user, elements of the user's record are sent from the HLR in the home system to the VLR in the non-home system. The home HLR registers the location of the user so that incoming calls can be routed.

Slides 52 – 53 Handover Types

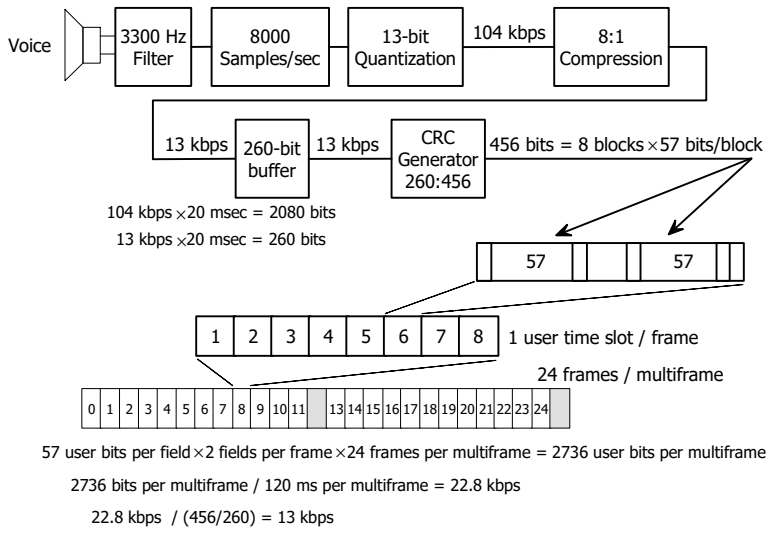
A GSM channel is defined by a frequency pair (upstream and downstream) used by the MS to communicate with the BTS. GSM defines four types of handover:

Intra-cell	The MS changes frequencies to avoid interference.
Inter-cell / Intra-cluster	The MS moves between cells within the control of one BSC (cluster).
Inter-cluster / Intra-MSC	The MS moves between cells controlled by different BSCs (clusters) and the MSC controls the handover.
Inter MSC	The MS moves between cells controlled by different MSCs

The GSM inter-BSC handover procedure is shown in slide 52. The MS monitors the received signal power from neighboring cells and the cluster determines that a handover is necessary. The request is forwarded to the MSC, which requests the handover from a new cluster. A cell in the new cluster activates a channel, passing the ACK of the request back to the new cluster, to the MSC to the old cluster, the old cell, and the MS. The MS begins to operate on the new channel, and confirmation of completion is sent to the MSC, old cluster and old cell.

Slide 54 GSM Voice Transmission Summary

The digital transmission of voice in GSM is as follows:



- The voice is band-limited to an analog range of 300 to 3300 Hz and sampled at a rate of 8000 samples per second (as in a Bezeq phone).
- The samples are rounded off to a 13-bit number (8192 levels instead of 256 levels on a Bezeq phone). 8000 samples/second × 13 bits/sample = 104 kbps.
- The digital stream is compressed at a ratio of 8:1, producing a 13 kbps stream.
- The bits are fed into a 260-bit buffer (one 20 msec sample). For each 260 bits a CRC error correction code of 196 bits is added, producing 456 bits. The transmission rate is now 456 bits / 0.020 seconds = 22.8 kbps.
- Each 456-bit block is divided into 8 blocks of 57 bits. A user time slot is constructed which includes control information and a pair of 57-bit blocks.
- Time slots from 8 users are combined into a frame.
- The GSM system transmits multiframes with 26 frame slots. Frame 12 and frame 25 are not used, so that the multiframe contains 24 frames, which contain 192 user time slots, which contain 2736 bits from any given user.

Slides 55 – 56 GSM Protocol Stack

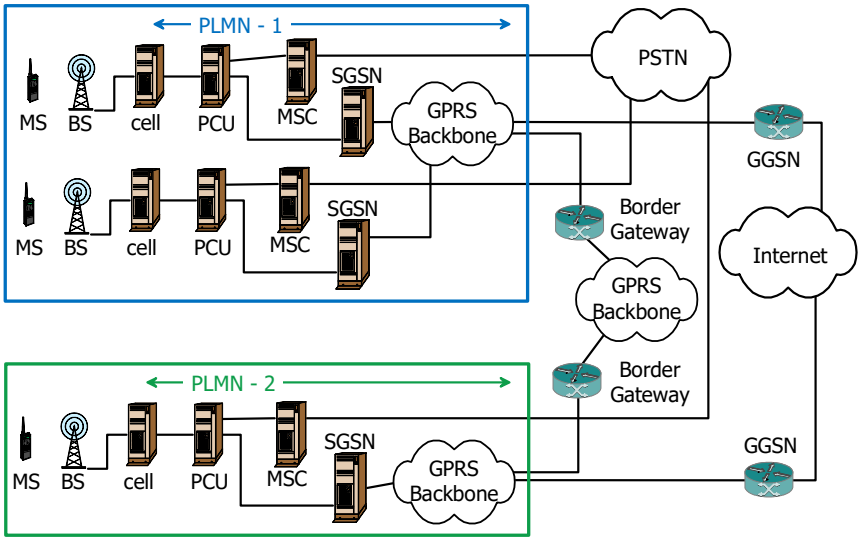
As seen on slide 55, the GSM protocol stack is quite complex. The top layers operate between the MS and the MSC for connection and mobility control. The BSC (cluster) and MSC run three protocols (BSSAP, SCCP, and MTP) to manage telephone call signaling.

The RMM protocol handles channel (frequency pair) allocation to the MS from the BSC. The SCCP signaling control connection between the MSC and BSC is converted to a BTSM signaling control connection between the BSC and BTS, and then to another RMM instance between the BTS and MS. The MTP that handles standard telephone signaling (ring indication, buys signal and so on) between the MSC and BSC (under control of the SCCP connection) is converted to LAPD, a standard signaling protocol for digital telephones.

The taxonomy of channel structures (slide 55) multiplexed over the GSM multiframes is similarly complex.

Slides 57 – 61 General Packet Radio Service (GPRS)

GPRS provides packet mode data access for over the GSM and 3G networks using an IP-based architecture. IP datagrams are separated from circuit mode voice traffic at the cluster by adding new functions to the BSC cluster controllers.



The **Packet Control Unit (PCU)** provides a packet mode function in the BSC to handle IP datagrams. Circuit mode voice/data are routed to MSC for forwarding to another MSC or PSTN (public switched telephone network, such as Bezeq). Packet mode data is routed to a **Serving GPRS Support Node (SGSN)** for forwarding to the Internet. In GSM the PCU to SGSN connection runs IP over Frame Relay, and in 3G it runs IP over ATM.

Mobility management for circuit mode voice traffic uses standard PSTN / PLMN routing, but packet mode traffic uses IP routing. The IP routing requires two new support nodes, SGSN and GGSN.

The **Serving GPRS Support Node (SGSN)** can be seen as a packet-switched version of the MSC. It handles packets to and from the MS and manages MS mobility, keeping track of the MS location and status in the mobile system. The SGSNs operate over a GPRS backbone jointly operated by cellular operators. MS to MS IP traffic can move over the GPRS backbone without exiting to the Internet backbone via a GGSN.

The **Gateway GPRS Support Node (GGSN)** provides an interface SGSNs to external IP networks, maintaining routing information for the SGSN currently serving a user. It is the exterior gateway for GPRS network. It runs DHCP to assign IP addresses to the MS and routes incoming IP datagrams to appropriate PCU. Traffic that leaves the GGSN to the Internet is pure IP traffic with no information about MS location or status. That information must be reattached to replies from the Internet entering the GGSN, so that the SGSN can route data to the appropriate PCU.

The **GPRS connection process** is shown in slide 61. The MS switches on and sends a GPRS attach request. User Registration (1) is required to associate a PLMN address (IMSI) with a Packet Data Protocol (PDP) address (a static or dynamic IP address). The BSC queries the HLR (2) for user authentication, and the HLR updates a VLR (3). Call Admission Control (CAC) determines the required network resources and grants resources if available (4). IP datagram delivery is now performed by hop-by-hop routing using routing tables in the GSN (GGSN or SGSN), by way of the

address conversion and VC mapping of the tunnel protocols. The GSN also handles compression and encryption.

Another 2.75G enhancement was **High Speed Circuit Switched Data (HSCSD)**, which permitted data-only switched circuit channels. Standard GPRS is a packet switched (packet mode) service, which only requires capacity when data is being transmitted (and so **billing is based on download volume**). However, HSCSD is a circuit mode service, like a voice call, and requires dedicated system capacity for the entire call (and so **billing is based on air time**).

3G GRPS now uses **High Speed Downlink Packet Access (HSDPA)**, a fast packet mode service permitting download rates up to 337 Mbps. Slide 72 lists a few methods used in HSDPA in the physical layer to achieve these rates.

Slides 64 – 66 SMS in GSM Architecture

Short Message Service (SMS) is a text-oriented service introduced in 2G cellular. Text is transferred in the control frames of regular messages, so that they require almost no capacity from the network operator.

The SMS system adds 3 new nodes to the GSM system. A **Short Messaging Entity (SME)** is any entity that can receive or send SMS, whether a fixed network element, an MS, or another service center. The **Short Message Service Center (SMSC)** stores and forwards SMS between any SME and an MS. Within the SMSC, the **Gateway Message Service Center (GMSC)** receives an SMS, interrogates the HLR for routing information, and delivers the SMS to the MSC for the destination SME. The **Interworking Message Service Center (IWMSC)** receives an SMS from the MSC and delivers it to a different SMSC for forwarding if necessary.

The stages of delivery of an SMS are shown in slide 66.

Slides 67 – 75 CDMA

Code Division Multiple Access (CDMA) is a commercial system originally developed by Qualcomm. The initial version (now called CDMAone) operates on 1G/2G frequencies and is not compatible with 3G CDMA 2000, which is a collection of incremental enhancements.

CDMA (and CDMA 2000) allocate a 25 MHz radio band in each direction (upstream and downstream). This band is band divided into 1.25 MHz RF channels allowing 20 RF channels per cluster. The system uses DSSS digital transmission (see slide 13 of presentation 1), transmitting 1.2288 Mcps (million chips per second) in each 1.25 MHz radio channel. The chips are used to produce a **code division multiplex** in which each user has a different code (chip sequences for the transmission of 1 bit). All users can transmit simultaneously using independent chip sequences — orthogonal (Walsh) codes or pseudorandom noise (PN) codes that can be unambiguously separated at the receiver.

Orthogonal codes are formed in an m-dimensional vector space with the inner product

$$\mathbf{U} \cdot \mathbf{V} = \frac{1}{m} \sum_{i=1}^{m} U_i \times V_i$$

The **m orthonormal basis vectors** for the vector space are written S_i for i = 1, 2, 3, ... , m and satisfy

$$S_i \cdot S_j = m \times \delta_{ij} = \begin{cases} 0, i \neq j \\ m, i = j \end{cases}$$

Any vector **T** in the vector space can be written in the form

$$\mathbf{T} = \sum_{i=1}^{m} t_i \times S_i$$

where the coefficients t_i can be found in the usual way from

$$t_i = S_i \cdot \mathbf{T} = S_i \cdot \frac{1}{m} \sum_{j=1}^{m} t_j \times S_j = \frac{1}{m} \sum_{j=1}^{m} t_j \times \left(S_i \cdot S_j \right) = \frac{1}{m} \sum_{j=1}^{m} t_j \times m\delta_{ij} = t_i$$

The **coding scheme** for the forward channel is:

- The basis vector S_i is assigned as a chip sequence to station i, for i = 1, 2, 3, ... , m.
- The base station chooses the coefficient $t_i = \begin{cases} -1 \text{ , transmits bit 0 to station i} \\ 0 \text{ , no data for station i} \\ +1 \text{ , transmits bit 1 to station i} \end{cases}$ for each user station i.
- The receiver in each MS receives the vector $\mathbf{T} = \sum_{i=1}^{m} t_i \times S_i$
- Station i finds the data bit from $t_i = S_i \cdot \mathbf{T}$ using its assigned code S_i.

Slide 69 shows an example of 4 stations with basis vectors (codes)

$$S_1 = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix} \quad S_2 = \begin{bmatrix} -1 \\ +1 \\ +1 \\ -1 \end{bmatrix} \quad S_3 = \begin{bmatrix} -1 \\ -1 \\ +1 \\ +1 \end{bmatrix} \quad S_4 = \begin{bmatrix} -1 \\ +1 \\ -1 \\ +1 \end{bmatrix}$$

so that

$$S_2 \cdot S_2 = \frac{1}{4}\left((-1)^2 + (+1)^2 + (+1)^2 + (-1)^2 \right) = 1$$

$$S_1 \cdot S_2 = \frac{1}{4}\left((-1)(-1) + (-1)(+1) + (-1)(+1) + (-1)(-1) \right) = 0$$

Slide 70 shows a transmission example for which Stations 1, 2, and 4 each receive the two-bit sequence 0 then 1, while Station 3 receives no data then 1.

Multiplying each code by its coefficient and adding the signals produces the total radio signal amplitudes (+3,-1,+1,+1) and (-4,0,0,0).

Slide 72 shows that applying the inner product to the two amplitude sequences and the basis vectors, each station is able to find the correct bit transmitted to it.

Orthogonal codes can be built as Walsh codes as shown on slide 73:

W_0 is a 1 bit and its complement W_0' is a 0 bit.

W_1 is a square matrix of 3 W_0 elements and a W_0'.

W_2 is a square matrix of 3 W_1 elements and a W_1'. This gives the 4 basis codes used in the example.

This can be continued — a W_N matrix has dimensions $2^N \times 2^N$.

For the **reverse channel** (MS to base station), Walsh codes are not practical because they require synchronization in time among many stations. Instead pseudorandom noise codes are

used, which produce "nearly orthogonal" codes. A pseudorandom Bernoulli sequence is equivalent to a sequence of m coin tosses, and produces a nearly equal number of (+1) and (–1) elements in each code. By the central limit theorem, the probability of a (+1) or a (–1) is given by

$$P_{-1} = P(-1) = \frac{1}{2} + \delta \qquad P_{+1} = P(+1) = \frac{1}{2} - \delta \qquad |\delta| < \frac{1}{\sqrt{m}}$$

If two codes, A and B, are produced from as pseudorandom Bernoulli sequences $C^{(A)}$ and $C^{(B)}$, then we expect that

for A = B

$$\frac{1}{m}\sum_{i=1}^{m} C_i^{(A)} \times C_i^{(B)} = \frac{1}{m}\sum_{i=1}^{m} [\pm 1]^2 = 1$$

while for A ≠ B

$$\frac{1}{m}\sum_{i=1}^{m} C_i^{(A)} \times C_i^{(B)} = \frac{1}{m}\sum_{i=1}^{m} \left[P_{+1} \times P_{+1} - P_{+1} \times P_{-1} - P_{-1} \times P_{+1} + P_{-1} \times P_{-1}\right] = 4\delta^2 < \frac{4}{m}$$

So, for m large enough, the inner product will give 1 for A = B, and give a much smaller number for A ≠ B ("nearly orthogonal").

CDMAone uses 64 orthogonal Walsh codes in the forward channel and a long PN sequence in the reverse channel.

Slides 76 – 81 2G to 3G Cellular Transition

Slide 76 shows the changes made between 2G GSM and **3G** (UMTS). Aside from name changes of nodes, 3G is designed for greater coordination between clusters, implemented by a direct interface between the Radio Network Controllers (RNC) that replace the GSM BSC cluster controllers.

Slides 77 to 81 show the handover process in 3G UMTS. The MS is operating in cell 1 and moving toward cell 4. The MS is connected to cell 1 and being monitored by cell 2 in the same cluster (RNC-1). The MS moves to cell 2 within RNC-1 and is moving between cell 2 and cell 3 in the new cluster (RNC-2). The MS is connected to cell 2 and cell 3, with cell 4 monitoring. RNC-1 is in control and coordinates with RNC-2. The MS is then connected to cell 3 and cell 4 under control of RNC-1. Finally, the MS connects to cell 4 under control of RNC-2, with cell3 monitoring.

Slides 82 – 85 4G Cellular

The planning for 4G cellular began in 2000 in the ITU, at the same time that 3G was introduced. A major goal was NGN (next generation network) convergence with general support of mobility, with no distinction between voice and data. The principal objectives were state as:

- Higher network capacity than 3G
- Higher spectral efficiency (higher bit rates (bps) per analog bandwidth (Hz) and higher bps / Hz / per cell site)
- 100 Mbps for any moving client and 1 Gbps for any stationary client
- 100 Mbps between any two points in the world

- Smooth handoff across all heterogeneous network types
- Global roaming across multiple networks
- QoS for multimedia support — audio, HDTV, streaming
- Interoperability with existing wireless standards
- An all IPv6 packet switched network that eliminates circuit mode voice entirely (voice calls to be implemented as Voice over IP).

The world economic crisis of 2000 – 2001, the near economic collapse of 2007 – 2008, and international political turmoil since 2001 have left these goals largely unachieved. In the meantime, some new networking infrastructures have been introduced and manufacturers have been eager to label their products 4G. Recently the ITU withdrew its objection to the use of the term 4G to describe systems that do not meet the original 4G requirements.

In particular **Long Term Evolution (LTE)** is a 3.5G service marketed as 4G. LTE provides an improved radio interface and higher data rates, up to 299.6 Mbps downstream and 75.4 Mbps upstream. LTE also supports improved mobility, supporting an MS moving at 500 km/h.

Voice over LTE (VoLTE) is a packet switched voice system that replaces the circuit switched voice network. It provides enhanced voice encoding as a VBR stream with circuit-switched fallback (CSFB) if the packet mode system does not provide adequate performance. As an intermediate migration path, most cell providers have been installing data over LTE together with a standard circuit switched voice network.

As is the 2G to 3G transition, a number of network elements have been upgraded in LTE. The 3G UTRAN is replaced with the Evolved UTRAN (e-UTRAN). Similarly, the eNodeB replaces the UMTS NodeB in each e-UTRAN.

The general network architecture is called the **Evolved Packet Core (EPC)**. It includes the Mobility Management Entity (MME) that replaces the GSM VLR for management of roaming, and the Home Subscriber Server (HSS) that replaces GSM HLR for management of devices in their home networks.

As an all-IP network, no circuit mode switches are implemented and the hierarchical network of MSC / SGSN / nodes in GSM/UMTS is absent. The Serving Gateway (SGW) acts as an IP router, routing and forwarding user data packets within the cellular network. The Packet Data Network Gateway (PGW) routes and forwards user data packets to the Internet, replacing the GGSN in GSM/UMTS.

Slide 86 5G Cellular

The general plan for 5G cellular specifies a full redefinition of the cellular system as an abstract and global Software Defined Network (SDN). It is conceived as a Service Based Architecture (SBA), in which network functions are defined as stateless services rather than hardware functions. The functions are implemented through extensive virtualization. In general, networks will be designed by standards of software engineering.

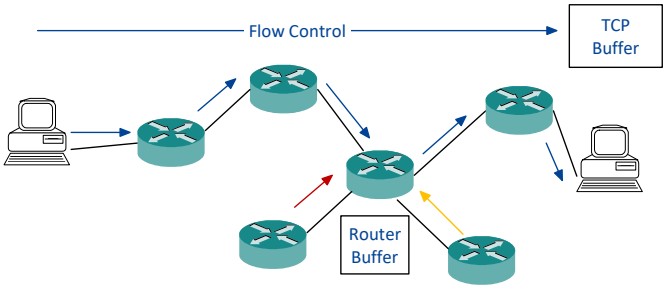
Presentation 5

Congestion and Flow Control in TCP

Congestion and flow control are basic and essential functions for reliable transport, and for most use cases are implemented in the TCP protocol. These functions add overhead to network operation and in many circumstances degrades the transmission throughput unnecessarily.

Slides 2 – 5 Flow Control and Congestion Control

The hosts in a TCP session maintain buffers filled and emptied by the application layer. Internet routers maintain buffers of IP datagrams waiting to be forwarded.



Recalling the basic definitions,

- **Flow control** permits the sender to avoid overflow of the receiver's buffer.
- **Congestion control** permits all senders to avoid overflow of intermediate network buffers, especially router buffers.
- **Buffer fill rate** is the number of bytes per second arriving at a buffer from the network.
- **Buffer empty rate** is the number of bytes per second leaving the buffer to the network or to the application layer at the destination host.
- **Buffer fill time** is the time to overflow an empty buffer, given by

$$T_{overflow} = \frac{\text{buffer size}}{\text{buffer fill rate} - \text{buffer empty rate}}$$

From queuing theory we learn that buffer level and latency (waiting time for an arriving packet in a buffer) depends on **buffer utilization** defined as

$$\rho = \text{Utilization} = \frac{\text{arrival rate}}{\text{empty rate}}$$

$$\text{Latency} = \frac{1}{\text{empty rate} - \text{arrival rate}} = \frac{1}{\text{empty rate}}\left(\frac{1}{1-\rho}\right)$$

$$\text{Buffer Level} = \text{Latency} \times \text{arrival rate} = \frac{\rho}{1-\rho}$$

Because long buffer delays can cause timeouts and retransmissions in TCP, the data throughput begins to decline at about 85% utilization.

Slides 6 – 9 TCP Flow Control

Flow control in TCP is managed by a "sliding window" method. The sender maintains a **source window**, the maximum number of TCP segments it is permitted to transmit before stopping to wait for ACKs. The initial source window is set by a system policy and is the maximum number of bytes permitted "in flight" (send and not yet ACKed). The source window is adjusted by mechanisms of congestion and flow control.

The receiver maintains a **destination window**, which is the number of bytes that the receiver can accept, and is determined by the available space in the receiver buffer. At any given time, the receiver buffer level is adjusted as the previous level plus the number of arriving bytes minus the number of bytes read by the application layer. If the application reads too slowly, then TCP must decrease its destination window.

To coordinate the flow rate the sender and receiver use a **sliding window**. The receiver uses the **window field** in the TCP header to report the number of bytes it will accept. The sender must adjust its source window — the receiver will discard any received bytes beyond its window size.

Slide 7 shows an example of TCP flow control.

- Before transmission (no bytes in flight), the sender sets a policy-based source window of 64 KB, and the receiver (with empty buffer) sets its destination window to 8 KB (its actual buffer size).
- The sender transmits two segments of 2 KB each. The in-flight counter rises to 4 KB and the buffer level at the receiver also rises to 4 KB. The receiver resets its destination window to 4 KB, and reports this number to the sender in the window field of the ACK.
- After the first ACK, the sender in-flight counter returns to 0 (all sent bytes were ACKed), and adjusts its source window to 4 KB (as reported by the receiver).
- The sender transmits another 2 KB, adjusting its in-flight counter and source window, and the receiver adjusts its buffer level and destination window.
- The application then reads 4 KB from the buffer. The receiver readjusts its buffer level and destination window, and reports the new destination window in the ACK.
- The sender now transmits 6 KB, adjusting its in-flight counter and setting its source window to 0. The receiver buffer is now full and the receiver reports a receive window of 0. The in-flight counter is 0 but the sender cannot transmit because the source window is 0 as part of the flow control mechanism.
- When the application reads 4 KB of data, the receiver repeats the previous ACK to report a receive window of 4 KB. But this ACK is lost and the sender does not know that it can transmit. The sender waits a certain interval (**persist timeout**) and then sends one byte to check the receiver. The receiver sends an ACK of the 1 byte and again reports a destination window of 4 KB (minus 1 byte). The sender is now free to continue transmitting.

Slide 8 again shows the deadlock problem solved by the persist timeout. Slide 9 shows another bug, the **silly window problem**, in which a low receiver application leads to a large number of short segments and ACKs (more header than data). In the Nagle algorithm, the sender buffers data, waiting until it can send a large segment. This works badly with text-oriented transmission like Telnet. An alternative is to keep the window size at 0 until the destination window is a reasonable size.

Slide 10 TCP Congestion Control

End-to-end congestion control in TCP is far more complex than flow control. The transport layer (layer 4 in the Internet model) is meant to ignore the details of the Network layer (layer 3) and receives no feedback from the intermediate network nodes. Therefore congestion control in TCP is based on heuristic estimates based on data the host receives.

In **slow-start**, the sender begins the session with a low transmission rate and increases the rate exponentially until timeouts begin. In order to avoid waiting for a timeout (with no transmissions if the in-flight is at maximum), a **fast retransmit** mechanism triggers retransmission after 3 duplicate ACKs (dupACKs), which usually means that a packet was lost. **Congestion avoidance** moderates the growth of the transmission rate after duplicate ACKs.

Slides 11 – 12 Slow-Start

In the slow start method used by current TCP implementations, the source window is called a **congestion window** and associated with the internal sender variable **`cwnd`**. The congestion window is initialized to the maximum segment size (MSS), which is the largest number of bytes that can be sent in one TCP segment (packet of header and data). The sender can thus send one MSS and then must wait for an ACK before it can send more data.

The **round trip time (RTT)** is the average time between the transmission of a segment by the sender and the arrival of the ACK at the sender. Since the sender starts by sending 1 MSS and waiting RTT seconds for the ACK, the average data rate is 1 MSS / RTT.

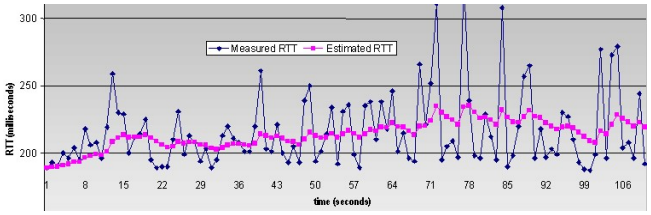
The send can raise the cwnd but the maximum cwnd is the destination window reported by the receiver. In slow start cwnd grows exponentially. On each ACK that arrives, the sender adds the size of ACKed data to the previous cwnd until cwnd reaches its maximum value. If there is congestion then at some value of cwnd the data rate will be too high, packets will be lost, and there will be a timeout while waiting for the required ACK. On a timeout, the cwnd is reset to 1 MSS.

This means that under conditions of congestion, the data rate will begin well below the capacity rate, grow exponentially until it is too high and causes a timeout, and then drops back below the capacity rate. On average, the **utilization** will be about half of the practical capacity under congestion.

In order to set the timeout interval (**RTO**) to a reasonable value, the sender must monitor the RTT of each ACK. The method of determining the RTO is shown on slide 12. The initial RTO is 3 seconds. After the first measured RTT, the RTO is set as

```
RTO = max(1 sec, SRTT + max(G, 4 * RTTVAR))
```

where SRTT is the measured RTT, G is a fixed granularity of 0.5 seconds, and RTTVAR is an estimate of the variance of RTT given by the measured RTT. After each measurement of subsequent values of RTT, the RTO is reset by the weighted average of old and new values shown in slide 12. The pink line in the graph shows the averaging outcome.



Slide 13 Fast Retransmit

In order to improve underutilization in the TCP slow start method, **fast retransmit** sets the RTO to be much longer than the average RTT and begins lost packet retransmission when 3 duplicate ACKs (**dupACKs**) are received with the same ACK number.

In the example, 5 segments are transmitted, each with 100 bytes of data so that the SEQ number rises from 100 to 500. After the first segment, the receiver sends ACK = 200, which acknowledges all data up to 199 and states an expectation to receive SEQ = 200. But the segment with SEQ = 200 is corrupt. When the segment with SEQ = 300 arrives, the receiver is still expecting SEQ = 200 and responds to the out-of-order segment by repeating the ACK of 200. Similarly, when the segments with SEQ = 400 and SEQ = 500 arrive, the receiver is still expecting SEQ = 200 and again responds by repeating the ACK of 200.

After receiving 3 dupACKs (4 ACKs) for 200, the sender retransmits the missing segment with SEQ = 200. Here the behavior of the receiver is implementation dependent. The receiver is permitted to discard all segments that arrive out of order (in this case, the receiver is expecting SEQ = 200 so a segment with SEQ = 300, 400 or 500 can be deleted). But most receivers (with a large enough buffer) will save the out-of-order segments. In the example, the receiver buffered the last 3 segments, so when the retransmitted segment with SEQ = 200 arrives, it sends a cumulative ACK with ACK = 600, which means that all previous data was received. In this case, the sender can transmit SEQ = 600 and does not need to retransmit the segments that arrived out of order.

Slide 14 Congestion Avoidance

Another way to avoid underutilization in the TCP slow start method is by moderating the exponential growth. The sender defines a new parameter called **slow start threshold** (**`ssthresh`**). As long as the **`cwnd < ssthresh`**, the data rate grows exponentially, by adding the number of ACKed bytes to the previous cwnd value. When **`cwnd > ssthresh`**, the data rate grows linearly, by adding 1 to the previous cwnd value.

Another modification called **Fast Retransmission** is that when there are 3 dupACKs, ssthresh is reset to half the previous cwnd and cwnd set to the new ssthresh + 3 (instead of being reset to 1). Only on a timeout is the cwnd reset to 1, and the RTO is doubled to avoid future timeouts.

On the 3rd dupACK, the sender retransmits the last unACKed packet and then sets

```
ssthresh ← cwnd / 2
cwnd ← ssthresh + 3
```

The sender waits for 1 RTT to see if the missing packet is ACKed.

If there are more dupACKs (instead of ACK for the missing packet), then the sender performs cwnd++ for each dupACK and resends the missing packet.

Slides 15 – 23 TCP Sender with Reno

TCP Reno is the basis for most implementations of TCP in use today. It includes, slow start, fast retransmit, and congestion avoidance.

Adding congestion control to the simplified TCO sender, the simplified Reno sender becomes

```
// initialize variables and parameters
       SEQ = ISN + 1
       SendBase = ISN + 1
       InFlight = 0
       cwnd = 1 MSS
       Set ssthreshold large (local policy)
       RTO = timeout interval
on (new data from application)                  // Handle data to be transmitted
       Prepare data segment:sequence number = SEQ
       if InFlight < min{cwnd,SendWindow,RecvWindow)
       Pass segment to IP
       SEQ = SEQ + length(data)
       InFlight = InFlight + length(data)
       if !(timer running) timer = RTO
if (receive ACK = y)                            // Handle an arriving ACK with value y
       stop timer
       if (y > SendBase)                        // check that y is a new ACK
              dupACK = 0
              newACKs = y – SendBase            // number of bytes ACKed this time
              SendBase = y                      // number of bytes ACKed so far
              InFlight = InFlight – newACKs     // number of bytes waiting for ACK
              if (cwnd < ssthresh)              // below threshold
                     cwnd = cwnd + newACKs      // exponential growth
              else                              // above threshold
                      cwnd = cwnd + 1 MSS       // linear growth
              if (InFlight > 0) timer = RTO     // set RTO if still waiting for ACKs

       else                                     // ACK is not a new ACK
              dupACK++                          // count dupACKs
              if (dupACK = 3)                   // fast retransmit trigger
                     SEQ = SendBase = min{unACKed SEQ} // just 1 packet
                     resend                     // resend last unACKed segment
                     timer = RTO                // reset timer
                     ssthresh = cwnd / 2        // update threshold
                     cwnd = ssthresh + 3        // reduce cwnd
                     wait 1 RTT                 // wait for ACK of resent packet
              if (dupACK > 3)
                     cwnd = cwnd + 1 MSS and resend again     // to permit retransmissions
if (timeout)                                    // Handle a timeout
       SEQ = SendBase = min{unACKed SEQ}
       resend                                   // resend last unACKed segment
       ssthresh = cwnd                          // update threshold
       cwnd = initial cwnd = 1 MSS              // reduce cwnd
       RTO = 2 * RTO                            // update RTO
       timer = RTO                              // reset timer
```

Adding congestion control to the simplified TCO receiver, the simplified Reno receiver becomes

```
// initialize variables and parameters
```

```
        Set RecvWindow = receiver buffer size
        expected = Sender ISN + 1
        ack_buffer = 0
        ack_max (local policy: delayed ACK trigger)
        ack_delay = 250 msec (local policy: < 500 msec)
        Start ACK delay timer = ack_delay
if (ACK delay timer = 0 && ack_buffer > 0)      // Handle delayed ACKs to be sent
        Send ACK = expected with updated RecvWindow
        ACK delay timer = ack_delay             // reset ACK delay timer
        ack_buffer = 0                          // reset delayed ACK number
if (receive SEQ = x)                            // Handle received segment with SEQ value x
        if (x = expected && error-free)         // correct order and not corrupted
                expected = expected + length(data)       // prepare next ACK value to send
        if (NACK = 1)                           // If this segment repairs a previous error
                Send ACK = expected with updated RecvWindow     // send immediate ACK
                ACK delay timer = ack_delay     // reset ACK delay timer
                ack_buffer = 0                  // reset delayed ACK number
                NACK = 0                        // reset NACK flag
        else if (ack_buffer < ack_max)          // If delayed ACK buffer not full
                nextACK = expected              // prepare next ACK value to send
                ack_buffer++                    // update delayed ACK number
        else if (ack_buffer = ack_max)          // If delayed ACK buffer is full
                Send ACK = expected with updated RecvWindow     // send immediate ACK
                ACK delay timer = ack_delay     // reset ACK delay timer
                ack_buffer = 0                  // reset delayed ACK number
        else                                    // Handle received segment with error or bad SEQ value x
                Send ACK = expected with updated RecvWindow     // send immediate ACK of expected
                ACK delay timer = ack_delay
                NACK = 1                        // set error flag
```

Slides 20 to 23 show an example of Reno TCP congestion control. The SEQ number is 1 MSS times the segment number SEQm, and the ACK number is 1 MSS times the number of the ACK segment ACKm. The receiver sends a delayed ACK for every 4 segments received, unless the delayed ACK timer times out or there is an error. The ssthresh is 32.

In the first line, the cwnd is 1 and the sender transmits SEQm = 1, so the in-flight counter is 1. Then the receiver sends ACKm = 2 (SEQm = 1 was received and now expecting SEQm = 2). This is an ACK of 1 segment. The sender adds 1 (new ACKs) to cwnd an resets in-flight to 0.

Next, the sender transmits 2 segments (SEQm = 2 and SEQm = 3) and in-flight is 2. The receiver sends 2 new ACKs by writing ACKm = 4 (received SEQm = 2 and SEQm = 3, and now expecting SEQm = 4). The sender adds the 2 new ACKs to cwnd and resets the in-flight counter.

Now the sender and receiver enter a "**self-clocking**" condition. The cwnd number rises exponentially with each received ACK. But the sender receives another ACK for every 4 segments it transmits. So, when the sender transmits 8 segments at one time (segments 8 to 15), it first receives an ACK for 4 segments (8 to 11) and then another ACK for 4 segments (12 to 15).

At the bottom of slide 21, cwnd reaches the threshold value of ssthresh = 32. So, on slide 22, for each new ACK the sender raises the cwnd by 1.

On slide 22 the sender does not transmit new segments so the in-flight drops by 4 on each ACK. At the top of slide 23, with cwnd = 38 the sender can transmit 26 new segments but transmits 8 new segments (SEQm = 68, 69, ... , 75) and raises in-flight from 12 to 20. The receiver sends an ACK of 4 more segments, with ACKm = 60 (previously sent at the top of slide 3). The sender adds 1 to cwnd and removes 4 from in-flight.

But now the receiver again sends ACKm = 60 another 3 times (3 dupACKs). These carry 0 new ACKs. This means that segment SEQm = 60 was lost or corrupted, but the sender does not know if additional segments were lost. The sender updates ssthresh to cwnd / 2 = 39/2 = 19, and sets cwnd to ssthresh + 3 = 19 + 3 = 22. The sender retransmits segment SEQm = 60 and waits 1 RTT.

The receiver now sends an ACK with ACKm = 76 (all segments up to SEQm = 75 correctly received). This is a cumulative ACK, so the sender can continue transmitting new segments from SEQm = 76 (it does not need to retransmit segments 68 to 75). Because cwnd = 22 is larger than ssthresh = 19, the 16 new ACKs in the cumulative ACK add 1 to cwnd which rises to 23. Since the in-flight counter is 0, the sender can transmit 23 segments.

Slide 24 Reno: First Alternative Scenario

In this case, the sender again starts from cwnd = 1, first sending 1 packet and then sending 2 packets (2 and 3). But this time the ACK following the second group is only for 1 packet. The sender updates cwnd to 3 but since InFlight = 1 it can only send 2 new packets (SEQ = 4, 5). Following each of these packets the sender receives a dupACK for 3. Since there are only 2 dupACKs (and no more to come, since cwnd = InFlight and so the sender must wait), there is a timeout, returning cwnd to 1 and ssthresh to 3 (the old cwnd).

Slides 25 – 26 Reno: Second Alternative Scenario

In this case, the sender again starts from cwnd = 1, first sending 1 packet, then sending 2 packets (2 and 3), and then sending 4 packets (4 – 7). This time the ACK following the third group is only for 3 packets (4 – 6). The sender updates cwnd to 7 but since InFlight = 1 it can only send 6 new packets (SEQ = 8 – 13). Following the first 5 of these packets the sender receives a dupACK for 7. After the 3rd dupACK the sender retransmits the missing packet 7 and reduced ssthresh to 3 and cwnd to 3+3 = 6. Since InFlight > cwnd, the sender cannot transmit, but there are more dupACKs arriving from the last 7 packets sent. Each dupACK > 3 causes cwnd++ so when cwnd > InFlight the sender can transmit packet 7 again. This time the sender receives a cumulative ACK for all the sent packets.

Slide 27 Selective Acknowledgment Option

The **selective acknowledgment (SACK)** option in TCP permits a partial ACK for a series of segments with gaps caused by lost segments. This is an option negotiated between hosts when both are capable of supporting the service. SACK acknowledgements use standard values for the SEQ and ACK fields, but include option fields showing data bytes that have arrived correctly.

For example, suppose that the last ACK received by the sender was 5000, and then the sender transmits 8 segments of 500 data bytes/segment (a total of 4000 additional bytes). If all the segments arrive with no errors, then the receiver will send ACK = 5000 + 8 × 500 = 9000.

In case 1, the first 4 segments are received correctly and the last 4 are dropped (lost). The receiver returns a normal ACK = 5000 + 4 × 500 = 7000. After a timeout, the sender will resend the last 4 segments and there is no need for a SACK option field.

In case 2, the first segment is lost and 7 segments are correctly received. For each received segment, the receiver returns a segment with ACK = 5000 and a SACK option field. The option field contains the start and end ACK for one received segment. The first segment (with

SEQ = 5000) is lost. The second segment with SEQ = 5500 contains 500 bytes. The receiver sends a segment with ACK = 5000 (to signify that SEQ = 5000 is still expected), and an option field with start = 5500 and end = 5999 (500 bytes received). The third segment with SEQ = 6000 also contains 500 bytes. The receiver sends a segment with ACK = 5000 (to signify that SEQ = 5000 is still expected), and an option field with start = 5500 and end = 6499 (a total of 1000 bytes received).

Slides 25 – 26 Active Queue Management (AQM)

In standard buffer queue management, a full buffer causes the receiver to drop (delete) excess packets. At the sender, there is no ACK for the dropped packet, which causes dupACKs or a timeout. The dupACKs or timeout cause retransmission of the dropped packet, but also cause the transmission rate to be lowered through cwnd. Thus dropped packets are a signal of congestion.

In **Random Early Detection (RED)**, the router detects congestion early. When the buffer level reaches about 85% of capacity, the router starts to drops random packets. The sender sees dupACKs or a timeout, assumes congestion, and lowers the cwnd before the congested buffer is completely full.

The RED algorithm is shown in slide 26. The parameters are:

- max_p is the maximum probability (0.1 to 0.5) of marking a packet as dropped
- min_{th} is a minimum threshold for the buffer queue, approximately 5
- max_{th} is a maximum threshold for the buffer queue, approximately 30
- p_b is a scaled marking probability found as $p_b = max_p \times (\text{average buffer level} - min_{th}) / (max_{th} - min_{th})$
- p_a is another scaled marking probability found as $p_b / (1 - \text{count} \times p_b)$, where count is the number of consecutive that were packets dropped

For each packet arrival,

- The buffer calculates avg = average queue size
- If $min_{th} \leq avg < max_{th}$ then the buffer calculates the probability p_a and marks the arriving packet for drop with probability p_a.
- If $max_{th} \leq avg$ then the buffer marks the arriving packet for drop with probability 1.

Notice that if the average buffer level is between the upper and lower thresholds, then $0 < (\text{average buffer level} - min_{th}) / (max_{th} - min_{th}) < 1$ so $0 < p_b < p_{max}$.

If p_b is stable, $1 - \text{count} \times p_b$ decreases with each consecutive dropped packed, and so p_a will increase as the number of consecutive dropped packets goes up.

Slides 27 – 44 AQM with ECN

Explicit Congestion Notification (ECN) is a method for using the RED algorithm in buffers to alert TCP to lower the transmission rate. In ECN, the IP router predicts congestion using the RED algorithm, but it marks the packets without actually dropping packets. The router indicates congestion in the direction from the sender to the receiver in the IP header. At the receiver the

IP software agent alerts TCP about the congestion, and TCP indicates congestion to the sender in the header of the TCP ACK.

Slide 28 shows the ECN fields in the IP datagram header. Two ECN bits are used to indicate 4 conditions:

00	Not ECN capable	Set for retransmissions to indicate that this packet cannot be randomly dropped
01	ECT(0) — ECN Capable Transport (0)	The sender alternates 01 and 10 to permit error control on the ECN mechanism
10	ECT(1) — ECN Capable Transport (1)	
11	CE (Congestion Experienced)	Set to indicate congestion in the direction of the IP datagram

Slide 29 shows the NS, CWR, and ECE flags in the TCP header. TCP uses the ECE flag to indicate congestion in the opposite direction, and uses CWR to allow the sender to acknowledge receipt of the ECE flag. The NS bit is used for error checking.

Slide 30 shows the setup of ECN in the TCP initial handshake. To use ECN, the SYN packet has ECE and CWR set (the IP packet sets ECN = 01), the server responds with ECE set in the SYN-ACK packet (the IP packet sets ECN = 10).

Slides 31 to 41 show the indication of congestion:

- Slide 31 shows no congestion. In TCP, ECE = CWR = 0, and in IP, ECN = 01 (ECT).
- Slide 32 shows no congestion in the reverse direction.
- Slide 33 shows the onset of congestion. In TCP, ECE = CWR = 0. In the IP header from the host to the router, ECN = 10 (the alternate ECT). But the router is starting to experience congestion and changes ECN to 11 (CE) in the IP header.
- Slide 34 shows the reverse direction. The TCP header now sets ECE = 1 CWR = 0 to report congestion to the sender TCP agent. The IP headers are set to ECT because there is no congestion in the reverse direction.
- Slide 35 shows that the sender lowers cwnd and sets CWR = 1 to acknowledge the congestion indication from the receiver side. ECE is 0 because there is no congestion to report in the reverse direction. In the IP headers, the first is ECT (no congestion from sender to router) and the second is CE (continued congestion from router to receiver).
- Slide 36 repeats the conditions of slide 34 (because the congestion continues).
- Slide 37 repeats the conditions of slide 35 (because the congestion continues).
- Slide 38 repeats the conditions of slide 36 (because the congestion continues).
- Slide 39 shows that the congestion has been cleared. The CWR is still 1 to acknowledge the previous congestion indication from the receiver side. This time the router sets the ECN bits to ECT (no congestion).
- Slide 40 shows the reverse direction with no congestion reporting. The TCP header has ECE = CWR = 0 and the IP headers have ECN set to ECT.
- Slide 41 shows that the congestion has been cleared, with ECE, CWR and ECN cleared. The sender can now raise the cwnd.

Slides 42 to 44 show performance results of RED and ECN. Goodput is the throughput of good packets (actual received data over TCP). Slide 42 shows that goodput drops with the number of

flows (TCP sessions running through the same router buffer) for RED and ECN with $p_{max} = 0.1$, but improved goodput for ECN with $p_{max} = 0.5$. Slide 43 shows delay, which is approximately the same for RED and ECN, but dependent on flow types (robust flows, such as large data transfers, are relatively insensitive to congestion, while fragile flows, such as video streaming, may stall during congestion). Slide 44 shows the dependence on RED parameters, with ECN outperforming pure RED.

Slides 45 – 47 ECN Nonce

The NS bit in the TCP header is used to detect errors in the ECN mechanism. Suppose that an unscrupulous or poorly implemented receiver clears the ECN-Echo signal (CWR) so that no congestion signals are ever sent to the sender. This can gives that receiver an advantage over connections that behave properly because it never warns the sender to lower the data transmission rate through cwnd.

To detect cheating, the NS bit (nonce) is set according to the history of the ECT signals. The sender and receiver keep a cumulative sum modulo 2 of all received nonces for all ACKed packets. The NS bit is this running sum (or 0 if the CE bit was set).

Slide 46 shows an honest receiver.

- NS is initialized to 1 in the SYN-ACK handshake.
- SEQ_1 arrives with ECT(0) so the nonce is (1 + 0)%2 = 1.
- SEQ_2 arrives with ECT(0) so the nonce is (1 + 0)%2 = 1.
- SEQ_3 arrives with ECT(1) so the nonce is (1 + 1)%2 = 0.
- The ACK of SEQ_3 is sent with NS = 0 and the sender sees the correct nonce for SEQ_3.
- SEQ_4 arrives with ECT(0) so the nonce is (0 + 0)%2 = 0.
- SEQ_5 is sent with ECT(1) but arrives with CE so the nonce is 0.
- The ACK of SEQ_6 is sent with NS = 0 and with ECE = 1, so the sender knows that there was a CE that set the nonce to 0 (otherwise it would have been 1).

Slide 47 shows a cheating receiver.

- NS is initialized to 1 in the SYN-ACK handshake.
- SEQ_1 arrives with ECT(1) so the nonce is (1 + 1)%2 = 0.
- SEQ_2 arrives with ECT(0) so the nonce is (0 + 0)%2 = 0.
- SEQ_3 is sent with ECT(1) but arrives with CE so the nonce is 0.
- The receiver does not set ECE = 1 (to cheat) and so it does not automatically set NS = 0. The receiver must guess what value of ECT to use for the nonce for SEQ_3. The ACK of SEQ_3 is sent with NS = 0 but the sender sees that the correct nonce should have been 1.
- The receiver will continue to guess the running sums and will be wrong most of the time.

Presentation 6

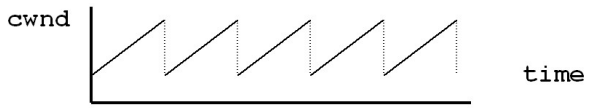
TCP over Wireless

The TCP protocol was designed many years before the introduction of WiFi and cellular networking, and its methods of congestion and flow control are based on assumptions relevant to wired transmission. In this chapter we discuss the implications of TCP for wireless and mobile networks.

Slides 2 – 17 Classic TCP-Reno

TCP-Reno is the basis for the **standard TCP implementation** in use today. In ideal operation, the number of segments in-flight (transmitted and waiting for ACK) is equal to cwnd (the congestion window, the maximum number of bytes that may be sent without stopping to wait for an ACK, measured in units of the MSS). The receiver generally returns a cumulative ACK for each 2 to 4 segments, and so with cwnd > 0, there is near-continuous transmission most of the time.

Given the finite-sized buffers in intermediate routers and the collision avoidance scheme, by which cwnd increases by 1 until a buffer loss divides cwnd in half, the expected behavior of cwnd over time is as shown in the graph:



The **RTT (Round Trip Time)** is the sum of four factors: the data transmit time, the time the segment spends in send buffers, the ACK transmit time and the time the ACK spends in ACK buffers.

The **buffer time** is approximately equal to the typical service time per segment × the buffer level.

Because these factors vary, the RTT is a random variable that can rise and fall sharply.

In TCP a **timeout** occurs when the buffer level is high, so the buffer time is large and the total RTT is larger than the RTO (retransmit timeout). For the TCP sender, the packet is considered lost. A packet arrives at the receiver **out-of-order** (OOO) if the buffer level is high enough for packet k that its RTT is larger than the RTT for packet k+1 (so that packet k is considered lost, but packet k + 1 is ACKed before its RTO runs out). The receiver will send a cumulative ACK if the OOO packet was not lost (for example if packet 16 arrives before packet 15 because over different data paths).

A **buffer error condition** occurs whenever a buffer level causes RTT > RTO for time interval T_{error}. The system sees an **isolated error** if T_{error} is shorter than the time between packets. In this case, the buffer empties before next packet arrives and there is no need to lower the transmission rate. **Congestion** occurs when T_{error} is longer than the time between packets, so that the problem causes multiple buffer errors and it is necessary to lower the transmission rate to reduce buffer errors.

The average transmission rate is $R = \left\langle \frac{\text{packets sent}}{\text{time to ACK}} \right\rangle = \left\langle \frac{\text{cwnd}}{\text{RTT}} \right\rangle$ and so R decreases if RTT increases or cwnd decreases. In an **isolated buffer error**, T_{error} is shorter than the time between packets so that RTT increases causing R to decrease during the time interval $T_{error.}$ In this case, the buffer corrects itself and there is no need to change cwnd. In **congestion**, T_{error} is longer

than the time between packets, so that even though RTT increases and causes R to decrease, the buffer is still too full. To address this, senders must decrease cwnd to decrease R even more.

Congestion control depends on an accurate estimate of RTT. The standard method for determining RTT in TCP is shown on slide 34. It turns out that this method tracks RTT too slowly. By taking an average that includes all past data, this measurement does not predict local congestion well.

Slide 6 shows the typical behavior of TCP-Reno when there is no congestion. The cwnd grows by 1 after each ACK, and on the next transmit cycle 1 additional segment can be transmitted. The value of in-flight in this example is stable at 3 (less than cwnd) because the receiver sends an ACK after 3 segments.

Slide 7 shows a numerical simulation of the variables SEQ, ACK, and cwnd over time. In the slow start phase ACK and cwnd overlap.

- At time = 0, cwnd = 1 so that SEQ = 1.
- At time = 0.5 the receiver returns ACK = 2 (SEQ = 2 is expected), which is 1 new ACK, so cwnd is raised to 2. Two segments are transmitted (SEQ = 2 and SEQ =3), and so SEQ = 3.
- At time = 1.0 the receiver returns ACK = 4 (SEQ = 4 is expected), which is 2 new ACKs, so cwnd is raised to 4. Four segments are transmitted (SEQ = 4, ... , 7), and so SEQ = 7.
- At time = 1.3 the receiver returns ACK = 7 (SEQ = 8 is expected), which is 4 new ACKs, so cwnd is raised to 8. Eight segments are transmitted (SEQ = 8, ... , 15), and so SEQ = 15.
- At time = 1.6 the receiver returns ACK = 12 and then ACK = 16, which are 8 new ACKs, so cwnd is raised to 16. Sixteen segments are transmitted (SEQ = 16, ... , 31), and so SEQ = 31.
- At time = 1.9 the receiver returns ACK = 20, ACK = 24, ACK = 28 and ACK = 32, which are 16 new ACKs, so cwnd is raised to 32. Thirty-two segments are transmitted (SEQ = 32, ... , 63), and so SEQ = 63. At cwnd = 32 the sender enters the congestion avoidance phase. ACKs arrive 4 at a time, and for each ACK, cwnd rises by 1.
- The sender transmits SEQ = 63 + 33 = 96, receives 4 ACKs at a time, and stops.

The latency (time from initial SEQ to final ACK) for the transmission of 96 segments is 2.77 seconds. The utilization (segments transmitted as a proportion of possible segments transmitted with cwnd $\rightarrow \infty$) is 34.2% and because there were no errors, this is also the goodput.

Slide 8 reviews the model of **steady state collision avoidance**. Assuming that the cwnd is fully utilized, n new ACKs reduces in-flight by n and raises cwnd by 1. The sender will then transmit n+1 new segments. The data transmission rate is just (cwnd × segment size) / RTT, which will decrease if RTT (round trip time) grows due to congestion or if cwnd is reduced by the sender. Decreasing the transmission rate is intended to give the buffers time to empty and reduce the congestion. The fill level of the buffer is indicated o the right side of the slide.

Slide 9 shows transmission under congestion (solid lines) compared with the congestion-free case of slide 4 (dotted lines). Congestion delays an ACK from time 2.2 to time 2.5. The total latency increases to 3.70 (0.3 is the actual excess delay and 0.63 is the delay in raising cwnd from a more restricted value). The utilization drops from 34.2% to 25.6%.

Slide 10 shows a typical steady state condition. The cwnd grows as usual (exponentially or linearly) until a segment is lost (SEQ = 9, 29, 49, 69, 89). After 3 dupACKs, the cwnd is cut in half and transmission continues. The drops in the red line show the retransmissions, which return the SEQ variable to the last ACKed segment. In this case, the total latency is 5.32, which is 92%

higher than the error-free case. The utilization drops to 20.8% and since some of those transmissions required retransmission, the goodput is only 17.8%.

Slides 11 and 12 show a loss of one segment in detail. With cwnd = 4, the first of 4 segments (SEQ = 7000) is lost. Because 3 additional segments follow, the sender receives 3 dupACKs, cuts cwnd in half, and sends 1 segment during RTT. The ACK raises cwnd to 3 and the sender transmits 3 more segments. Slide 9 shows the values of cwnd, SEQ, and ACK over time for this case.

Slide 13 shows the case of multiple successive segment losses. Because the sender cannot transmit additional segments, there are no dupACKs to indicate lost segments, and the sender cannot act until the timeout, which returns cwnd to 1 to begin retransmission. In this case, the long latency is required to reduce the transmission rate and permit the buffer to recover from congestion.

Slide 14 shows that a similar excess latency can be caused by the loss of a single packet early in the transmission when cwnd is small. SEQ = 1 is sent and ACK = 2 arrives at time 0.25. The cwnd is now 2 so that SEQ = 2 and SEQ = 3 are transmitted. At time 0.50 ACK = 3 arrives (acknowledging SEQ = 2) but no ACK arrives for SEQ = 3. Because the sender cannot transmit additional segments, there are no dupACKs for the lost segment and at time 4.0 the timeout returns cwnd to 1 and SEQ = 3 is retransmitted. If there are no additional errors, the transmission of SEQ = 96 and ACK = 97 will complete at time 6.87. Compared to the error-free case (slide 7) the excess latency is 48% higher and the utilization is only 14.1%. This shows that one early packet loss can be as costly as several packet losses in steady state. Slide 15 shows the very similar case of one lost packet retransmitted after 3 dupACKs, but followed by a loss of the retransmission.

Slides 16 and 17 summarize the operation of congestion control in TCP-Reno. The probability of a single bit error (which causes a single TCP segment loss) on a wired network is very small (around 10^{-5}) and so lost packets are interpreted as a sign of congestion. TCP-Reno has a moderate response to a packet loss, reacting after 3 dupACKs and lowering the cwnd by half. The probability of multiple successive packet errors on a wired network is extremely small, and so the more serious response to a timeout is considered necessary to allow the buffers to overcome the congestion condition.

Slides 18 – 21 The Trouble with Wireless

Despite the assumption is TCP, the probability of a single bit error in a wireless network, such as WiFi and cellular, is much higher than the bit error rate on a wired network. In addition, the probability of a long sequence of correlated errors is much higher, because errors are caused by external interference that may last over multiple segment transmission times. When TCP treats multiple successive segment losses as a sign of heavy congestion, it is usually wrong. This means that the overall utilization of wireless channels is kept unnecessarily low, when congestion is not a problem.

Transmission errors in wireless networks are caused by variations in the transmission medium and external interference. The conditions that cause refraction, reflection, and absorption can change with temperature and wind conditions, as well as with mobility of the wireless station. Sources or electromagnetic interference (other wireless users, electrical equipment, microwave ovens, and so on) may be active for seconds or minutes at a time, causing long sequences of correlated errors that do not occur on wired networks.

Slide 19 shows the problem of mobility drops — delays caused by mobility management. As the user moves from one wireless domain to another and additional node buffers are added to the system, the delay time for sending and receiving ACK messages may suddenly become longer than average, causing dupACKs and timeouts. This delay variation will be interpreted by TCP as a sign of congestion, leading to a drop in cwnd and loss of throughput.

Slide 20 shows the problem of ACK compression. Wireless links are usually asymmetric, with the upstream channel (user to base) transmitting much slower than the downstream channel (base to user). As a result the mobile device will operate with a larger buffer, leading to a longer buffer delay to accommodate the lower average transmission rate. In the example, the base sends three segments and the mobile device buffers several ACKs until a convenient time to transmit in the wireless system. The base station receives a larger than normal number of ACKs very quickly so it raises the cwnd very quickly. The RTT for the first ACK was long, but the remaining ACKs arrived soon after, so the average RTT is seen to be short. The base can then transmit a large number of segments very quickly, which may flood the mobile device and cause overflow in its receive buffer.

Slide 21 shows a problem in TCP caused by the use of the reliable MACA protocol in WiFi. MACA has overhead associated with the RTS, CTS and ACK_{MAC} packets. The time to handle this overhead may be as long at the TCP ACK time, and so RTT will be twice the RTT on a wired network. Because the long RTT is interpreted as congestion, the TCP agent will lower the average transmission rate unnecessarily.

Slides 22 – 23 Typology of Approaches to Wireless TCP

Several approaches have been proposed to modify TCP for use over wireless networks. The basic approaches can be divided into two groups:

- Split mode approaches were proposed in the 1990s, using standard TCP between the wireless base station and server, and specialized TCP between the base and mobile.
- End-to-end approaches have been proposed more recently. These are refinements to TCP-Reno for improved handling of non-congestion packet loss. They often require infrastructure-awareness — the use of IP and infrastructure layer information at the TCP layer (as was seen when ECN provides IP router information to TCP).

Within these groups, proposals can be divided into reactive and proactive congestion control:

- Reactive congestion control responds to a perceived congestion loss in the way that TCP-Reno cuts cwnd after 3 dupACKs.
- Proactive congestion control attempts to predict the actual available bandwidth from ACK arrival rates.

Slide 23 is an example of the case that many wireless TCP proposal address.

- With cwnd = 16 the sender transmits 16 segments, SEQ = 16 to SEQ = 31.
- 10 segments (SEQ = 16 to SEQ = 25) arrive correctly and the receiver returns ACK = 26. The sender reduces in-flight to 16 – 10 = 6 and raises cwnd to 16 + 10 = 26.
- 3 segments (SEQ = 26, 27, 28) are lost.
- 3 segments (SEQ = 29, 30, 31) arrive correctly. Because the receiver is still waiting for SEQ = 26, it returns ACK = 26 for each of these segments.

- The 3 dupACKs of 26 cause the sender to cut cwnd from 26 to 13 and retransmit SEQ = 26.
- The retransmission of SEQ = 26 arrives correctly, but now the receiver is still waiting for SEQ = 27 and sends ACK = 27. The sender adds 1 to cwnd and subtracts 1 from in-flight. But because cwnd > in-flight, and there has been no timeout for segments in-flight, the sender can assume that the ACK = 28 to ACK = 32 will arrive later. It transmits 9 more segments (SEQ = 32 to 40).
- The receiver responds to the 9 new segments by sending 9 dupACKs with ACK = 27.
- After 3 of these dupACKs, the sender cuts cwnd from 14 to 7 and retransmits SEQ = 27. The retransmission arrives correctly but the receiver is still waiting for SEQ = 28 and sends ACK = 28.
- But because cwnd > in-flight, and there has been no timeout for segments in-flight, the sender can assume that the ACK = 29 to ACK = 40 will arrive later. It transmits 1 more segments (SEQ = 41).
- The receiver responds to the new segment by returning a dupACK for ACK = 28, but it cannot send more dupACKs because no more segments arrive.
- Finally the sender has a timeout on SEQ = 28. It retransmits SEQ = 28.
- The receiver has buffered the out-of-order segments (SEQ = 29 to 41) and sends a cumulative ACK = 42. The sender can now continue transmitting.

Slide 24 New Reno

New Reno is a slight modification of TCP-Reno standardized by the Internet Engineering Task Force (IETF) as RFC 6582. The idea is that after 3 dupACKs the sender continues **fast recovery** until all in-flight segments are ACKed. This means that it retransmits all segments following the dupACK until it receives a cumulative ACK.

Slide 24 shows the same case as on slide 23 — the sender transmits 16 segments with but 3 segments (SEQ = 26 to 28) are lost. The receiver returns 3 dupACKs for ACK = 26, putting the sender into fast recovery mode. After receiving ACK = 27, instead of assuming that the remaining ACKs (28 to 31) will arrive later, the sender continues to retransmit SEQ = 27 and 28. When SEQ = 28 arrives, the receiver returns a cumulative ACK = 31, which ACKs all the segments in-flight. The receiver can now continue transmitting, with no loss of time caused by a timeout.

Slide 25 TCP-NCL (Non-Congestive Loss)

TCP-NCL was proposed in 2010 to improve wireless transmission based on a heuristic assumption: a single timeout could be caused by a bit error, but a timeout of the retransmitted segment is probably caused by congestion. The method is expressed in the slide as a decision tree. There are two separate timeout clocks, the Retransmission Decision (RD) whose timeout interval is the standard RTO, and the Congestion Decision (CD) with a longer timeout interval. A timeout of the RD causes a retransmission of the segment and starts the CD timer, but does not affect the value of cwnd (the transmission rate continues as it was). An ACK following the retransmission resets both the RD and the CD. If there is no ACK when the CD timeout occurs, then the congestion control mechanism returns cwnd to 1.

Slide 26 TCP Vegas

TCP Vegas was first proposed in 1995. Instead of changing the cwnd by a large amount after dupACKs or a timeout, it adjusts cwnd every few ACKs based on the changing RTT.

As in Reno, the slow start phase begins with cwnd = 1 and exponential growth of cwnd after each ACK. The sender defines an **expected throughput** given by the cwnd divided by the minimum value of RTT measured in the system (under no congestion). The sender also defines lower and upper **threshold values** α and β. Every few transmissions, the sender transmits a special segment, measures the RTT for that segment, and calculates the **actual throughput** as the size of the ACK received divided by the RTT for the special segment. The sender then calculates the variable **diff**, which is the difference between the actual and expected throughput. If diff < α then the sender raises cwnd by 1 on each ACK. If diff > β then the sender lowers cwnd by 1 on each ACK. If diff is between α and β then cwnd remains unchanged.

Fast recovery is also modified. On 3 dupACKs the sender retransmits the missing segment (as in Reno) but does not change the cwnd. This prevents packet errors from being treated like congestion. The sender also defines a Vega-timeout RTT. If the actual RTT is larger than this value then the sender begins retransmission after 1 dupACK.

Slides 27 – 28 TCP — Modified Fast Retransmission (MFR)

TCP — MFR is an attempt to count dupACKs as a measure of traffic that arrives but is not ACKed.

In Reno, as seen in slide 15, if cwnd segments are transmitted and segment n is lost then k dupACKs are returned, for some value of k. The sender retransmits after 3 dupACKs and waits for the ACK of segment n. But if the retransmitted segment is also lost, then there are no more dupACKs, and the sender must wait for a timeout on segment n.

In TCP-MFR, the sender receives dupACKs, assumes that only one segment was lost, and retransmits the missing segment. The sender also counts the k dupACKs that arrive and determines that k segments must have arrived (to trigger the return of k dupACKs). If segment n was lost and k segments arrived then k + 1 segments are accounted for, and cwnd – (k + 1) are unaccounted for.

If more than 3 dupACKs arrive then the sender increases cwnd by (k – 3) and sends more segments. If a cumulative ACK arrives then the retransmitted and additional segments all arrived. But if there are more dupACKs for segment n then some new segments arrived, but the retransmission of segment n was also lost. Now the sender retransmits segment n again with no timeout.

Slide 28 shows an example. The sender transmits 16 segments (SEQ = 16 to 31). All 16 segments arrive correctly except SEQ = 21. Segments SEQ = 16 to 20 arrive correctly and the receiver sends ACK = 21. The in-flight now drops from 16 to 11 and the cwnd rises from 16 to 21. Ten more segments (SEQ = 22 to 31) arrive and the receiver returns dupACKs for ACK = 21. After 3 dupACKs the sender drops cwnd to 13 and retransmits SEQ = 21 but it is also lost.

At this point, the sender has received 8 dupACKs for ACK = 21, so it knows that of the 16 segments transmitted, 5 were ACKed, at least 1 was lost and 8 must have arrived (but it does not know which segments). So, 2 segments are unaccounted for. So now, k – 3 = 8 – 3 = 5 which is bigger than the number of unaccounted segments, and the sender can add 5 to cwnd (now 18) and send 7 more segments (SEQ = 32 to 38). The receiver is still waiting for segment SEQ = 21 and sends dupACKs for ACK = 21. After 3 more dupACKs, the sender does not know if these are

new or delayed dupACKs. But then 6 more dupACKs for ACK = 21 arrive (9 dupACKs = 2 delayed dupACKs from the first segment loss and 7 new dupACKs from the second segment loss). The sender sees that k – 3 = 9 – 3 = 6 is larger than the 2 unaccounted segments and retransmits SEQ = 21 again. No timeout takes place.

Slide 29 Modified Congestion Control (MCC)

Modified Congestion Control (MCC) is an attempt to set cwnd according to measured capacity instead of just raining or lowering the value. The method defines a Bandwidth Estimate (BWE) based on measurements of RTT and the formula

$$\text{BWE} = \text{weighted average of} \frac{\text{bits ACKed}}{\text{RTT for bits_to_ACK}}$$

On ACK number k, the sender calculates the BWE for sample k and the weighted average

$$\text{BWE_s[k]} = \frac{\text{packets ACKed} \times \text{bytes per packet}}{\text{now} - \text{last_ACK_time}}$$

$$\text{BWE[k]} = (1-\beta) \times \frac{\text{BWE_sample[k] + BWE_sample[k-1]}}{2} + \beta \times \text{BWE[k-1]}$$

The value of cwnd is used as in TCP-Reno, but it is set after congestion as:

On 3 dupACKs, ssthresh is set as

$$\begin{aligned}
\text{ssthresh} &= \text{BWE} \frac{\text{RTT}_{\text{min}}}{\text{segment size}} \\
&= \text{average measured} \left(\frac{\text{bytes ACKed}}{\text{measured RTT}} \right) \times \left(\frac{\text{RTT}_{\text{min}}}{\text{bytes per segment}} \right) \\
&= \text{average measured} \left(\frac{\text{RTT}_{\text{min}}}{\text{measured RTT}} \times \frac{\text{bytes per segment} \times \text{segments ACKed in RTT}}{\text{bytes per segment}} \right) \\
&= \text{average measured} \left(\frac{\text{RTT}_{\text{min}}}{\text{measured RTT}} \times \text{segments ACKed in RTT} \right)
\end{aligned}$$

and cwnd is lowered to ssthresh (if cwnd > ssthresh).

On a timeout,

ssthresh = max{(BWE*RTTmin) / segment size, 2}

cwnd = 1

Slide 30 Proposed Solutions for WiFi Overhead

Slide 31 Freeze-TCP

Freeze-TCP is an attempt to handle mobility drops caused by handover delays and temporary signal fading by obstacles. The basic idea is that the mobile station monitors the radio signal strength from its base station and predicts impending disconnections caused by handover (from cell to cell) or obstacles. The Freeze-TCP receiver proactively sets the window field in the TCP header to 0, so that the TCP sender enters persist mode (it cannot transmit until the persist

timer runs out — see slide 7 in chapter 5). When the mobile station reconnects and establishes a strong radio connection, the receiver sends multiple ACKs of the last received packet. This prevents the exponential back off of cwnd in the sender (decrease of cwnd caused by dupACKs).

This method requires **infrastructure awareness** in the TCP layer, which must be exposed to networking details handled by lower layers, such as roaming and handover at layers 2 and 3, and radio signal quality at layers 1 and 2.

Slides 32 – 33 TCP-Jersey

TCP-Jersey defines Available Bandwidth Estimation (ABE), which is similar to the BWE used in MCC (slide 29), but measured in segments sent over a fixed time (instead of segments per second). Like, MCC the method uses ABE to determine expected capacity, instead of raising and lowering cwnd. Like Freeze-TCP (slide 31), the method requires explicit congestion notification (ECN) support in routers, making it infrastructure aware. The method defines three cases:

- A **congestion warning (CW)** occurs when an IP router marks a segment on a congested link and this information is passed to the TCP receiver, which returns an ECN warning to the TCP sender. If CW occurs with no packet loss then cwnd is set to ABE.
- On **3 dupACKs**, the sender retransmits the lost segment as usual. If the segment carrying the dupACK includes an ECN congestion warning, the cwnd is set to ABE. Otherwise, the sender assumes that the loss was caused by a data error and leaves cwnd unchanged.
- On a **timeout**, the sender retransmits the lost segment as usual. If the segment carrying the dupACK includes an ECN congestion warning, the cwnd is set to 1. Otherwise, the sender assumes that the loss was caused by a data error and leaves cwnd unchanged.

Slide 29 shows results of measurements on wireless networks. Jersey-TCP is seen to perform better (goodput at some bit error rate) than Reno, Tahoe (a predecessor of Reno), and Westwood, under conditions of congestion and no congestion. The problem with **TCP-Jersey** is that it requires modification to the TCP sender and receiver, as well as infrastructure awareness.

Slides 34 – 47 TCP Westwood

TCP Westwood combines the dupACK counting used in MFR (slide 27) with the BWE used in MCC (slide 29). Its performance is almost as good as TCP-Jersey but it has several advantages. First it only modifies TCP sender and so is not dependent on negotiation with TCP receiver. It is also not dependent on ECN support in routers or any other infrastructure issue.

In estimating the available capacity, Westwood counts both ACKs and dupACKs as successful traffic. On a packet loss it sets cwnd equal to the measured available bandwidth.

Westwood uses the Reno slow start and congestion avoidance modes to raise **cwnd**. As in MCC, on dupACKs and timeouts Westwood sets cwnd according to an estimate of available capacity. As in MFR, Westwood counts dupACKs in calculating the bandwidth samples. The basic idea is to count a dupACK as a segment that arrived (some segment must have arrived or there would not have been a dupACK) and then avoid double counting the ACK that confirms the actual arrival.

Slide 36 shows an example this dupACK counting.

- At time t_0 segment SEQ = 1 is sent.
- At time t_1 ACK = 2 (acknowledging SEQ = 1) arrives and the sender estimates the bandwidth as $1 / (t_1 – t_0)$. Segments SEQ = 2, 3 are sent.
- At time t_2 ACK = 4 arrives and the sender estimates the bandwidth as $2 / (t_2 – t_1)$. Segments SEQ = 4, 5, 6, 7 are sent. SEQ = 5 is lost.
- At time t_3 ACK = 5 arrives and the sender estimates the bandwidth as $1 / (t_3 – t_2)$. Segment SEQ = 8 is sent.
- At time t_4 ACK = 5 arrives and the sender estimates the bandwidth as $1 / (t_4 – t_3)$. The dupACK counter = 1.
- At time t_5 ACK = 5 arrives and the sender estimates the bandwidth as $1 / (t_5 – t_4)$. The dupACK counter = 2.
- At time t_6 ACK = 5 arrives and the sender estimates the bandwidth as $1 / (t_6 – t_5)$. The dupACK counter = 3. Segment SEQ = 5 is retransmitted (after 3 dupACKs) and new segment SEQ = 9 is transmitted.
- At time t_7 cumulative ACK = 10 arrives. The ACK jumps from 5 to 10, which provides 5 new ACKs. The sender already counted 3 dupACKs (following SEQ = 6, 7, 8) as segments that arrived, so the bandwidth estimate is based on 5 new ACKs – 3 dupACKs = 2 new segment arrivals. The sender estimates the bandwidth as $2 / (t_7 – t_6)$.

Slide 37 formalizes this counting method as follows:

`newACK = ACK – prevACK ;`	The number of packets ACKed by the new ACK
`// if (newACK = 1) do nothing`	No error condition — report 1 arrival

`if (newACK = 0)`
`  count++ ;`
`  newACK = 1 ;`
	A new ACK arrives but the ACK value is "stuck" on the old value, so that this ACK is a dupACK.
	Increment the dupACK counter and report this dupACK as 1 new arrival.

`if (newACK > 1)`
`  if (count >= newACK)`
`  count = count – newACK + 1 ;`
`  newACK = 1 ;`
	The ACK value advances, but not all packets that arrived at the receiver have been ACKed.
	Remove the ACKed packets from the dupACK count and count this dupACK as 1 new arrival.

`  else if (count < newACK)`
`  newACK = newACK - count ;`
`  count = 0;`
	All packets have been ACKed in the correct order.
	Report ACKed – counted (new ACKs) as the number of arrivals.
	Zero dupACK counter

`prevACK = ACK ;` `return(newACK);`	

Slide 38 shows an example of the counting method when there are no lost packets. The sender transmits SEQ = 1, SEQ = 2,3, SEQ = 4,5,6,7, and SEQ = 8,9,10,11,12,13,14,15 and the receiver returns ACKs 2, 4, 8, 12, and 16.

- On ACK = 2, newACK = 2 – 1 = 1, so the counter reports 1 new arrival.
- On ACK = 4, newACK = 4 – 2 = 2, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 2 new arrivals.
- On ACK = 8, newACK = 8 – 4 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 4 new arrivals.
- On ACK = 12, newACK = 12 – 8 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 4 new arrivals.
- On ACK = 16, newACK = 12 – 8 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 4 new arrivals.

Slide 39 shows an example of the counting method when one packet arrives in the wrong order. The sender transmits SEQ = 1, SEQ = 2,3, SEQ = 4,5,6,7, and SEQ = 8,9,10,11,12,13,14,15, but SEQ = 5 arrives before SEQ = 4. The receiver returns ACKs 2, 4, 4, 6, 10, 14, and 16.

- On ACK = 2, newACK = 2 – 1 = 1, so the counter reports 1 new arrival.
- On ACK = 4, newACK = 4 – 2 = 2, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 2 new arrivals.
- On the second ACK = 4, newACK = 4 – 4 = 0, so the counter increments the dupACK count and reports 1 new arrival.
- On ACK = 6, newACK = 6 – 4 = 2, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 1, the counter reports 1 new arrival and zeros the dupACK counter.
- On ACK = 10, newACK = 10 – 6 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 4 new arrivals.
- On ACK = 14, newACK = 10 – 6 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 4 new arrivals.
- On ACK = 16, newACK = 16 – 10 = 2, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 2 new arrivals.

Slide 40 shows another example of the counting method when one packet arrives in the wrong order. The sender transmits SEQ = 1, SEQ = 2,3, SEQ = 4,5,6,7, and SEQ = 8,9,10,11,12,13,14,15, but SEQ = 5, 6, 7 arrive before SEQ = 4. The receiver returns ACKs 2, 4, 4, 4, 4, 8, 12, and 16.

- On ACK = 2, newACK = 2 – 1 = 1, so the counter reports 1 new arrival.
- On ACK = 4, newACK = 4 – 2 = 2, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 2 new arrivals.
- On the second ACK = 4, newACK = 4 – 4 = 0, so the counter increments the dupACK count to 1 and reports 1 new arrival.
- On the third ACK = 4, newACK = 4 – 4 = 0, so the counter increments the dupACK count to 2 and reports 1 new arrival.
- On the fourth ACK = 4, newACK = 4 – 4 = 0, so the counter increments the dupACK count to 3 and reports 1 new arrival.

- On ACK = 8, newACK = 8 – 4 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 3, the counter reports 1 new arrival and zeros the dupACK counter.
- On ACK = 12, newACK = 12 – 8 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 4 new arrivals.
- On ACK = 16, newACK = 16 – 12 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 0, the counter reports 4 new arrivals.

Slides 41 and 42 show variations of the case in slide 40.

Slide 43 shows another example of the counting method when two packets arrive in the wrong order. The sender transmits SEQ = 1, SEQ = 2,3, SEQ = 4,5,6,7, and SEQ = 8,9,10,11,12,13,14,15, but SEQ = 9, 10, 11 and SEQ = 13, 14, 15 arrive before SEQ = 8 and SEQ = 12. The receiver returns ACKs 8, 8, 8, 8, 8, 8, 12, and 16.

- On ACK = 8 (not shown), newACK = 8 – 4 = 4, so the counter reports 4 new arrivals.
- On ACK = 8 (2^{nd} time), newACK = 8 – 8 = 0, so the counter increments the dupACK count to 1 and reports 1 new arrival.
- On ACK = 8 (3^{rd} time), newACK = 8 – 8 = 0, so the counter increments the dupACK count to 2 and reports 1 new arrival.
- On ACK = 8 (4^{th} time), newACK = 8 – 8 = 0, so the counter increments the dupACK count to 3 and reports 1 new arrival.
- On ACK = 8 (5^{th} time), newACK = 8 – 8 = 0, so the counter increments the dupACK count to 4 and reports 1 new arrival.
- On ACK = 8 (6^{th} time), newACK = 8 – 8 = 0, so the counter increments the dupACK count to 5 and reports 1 new arrival.
- On ACK = 8 (7^{th} time), newACK = 8 – 8 = 0, so the counter increments the dupACK count to 6 and reports 1 new arrival.
- On ACK = 12, newACK = 12 – 8 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 6, which is larger than newACK, the counter updates the count to 6 – 4 + 1 = 3 and reports 1 new arrival.
- On ACK = 16, newACK = 16 – 12 = 4, which is larger than 1, so the counter compares this value with the dupACK count. Since the count is 3, the counter reports 4 – 3 = 1 new arrival.

Slides 44 – 47 Westwind Congestion Control

Slide 44 shows the modifications of Reno made by Westwood. Reno slow start and congestion avoidance operate without change. As in MCC, on dupACKs and timeouts Westwood sets cwnd according to an estimate of available capacity, instead of lowering it arbitrarily. On a timeout, Westwood resets cwnd to 1, but set ssthresh to the bandwidth estimate.

Slide 45 compares performance of Westwood to Reno and SACK (slide 24 in chapter 5) and shows improved throughput at all error rates.

Slide 46 shows the values of cwnd and ssthresh over time for Westwood (left) and Reno (right). As expected, Westwood maintains higher cwnd (the main factor in throughput in the absence of heavy congestion). If congestion is low but there are frequent packet losses caused by bit errors,

then the behavior of cwnd in Reno will be similar to the steady state shown in slide 10 and repeated here.

In Westwood cwnd is set according to the measured throughput and not affected by bit errors. In particular, is useful to compare Reno and Westwood in **congestion avoidance**, in which case **cwnd = ssthresh**. Reno and Westwood raise cwnd linearly, adding 1 for each ACK. If there are periodic packet losses, then each time Reno cuts cwnd in half, leading to the steady state behavior shown in slide 46. But in Westwood, the bandwidth estimate (BWE) is based on the number of bytes ACKed (including dupACKs) and so ssthresh will continue to grow until it reaches a reasonable approximation to actual capacity. On a packet loss, the cwnd will not be cut (unless there is actual congestion that decreases ssthresh) and cwnd will keep growing linearly until it reaches the value of ssthresh that represents actual capacity.