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.

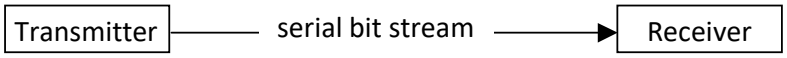
Telecommunications engineers study networking from the bottom up, in part because this is how networking infrastructure is designed and developed, and in part because that infrastructure is the locus of their expertise and responsibility. **Programmers** generally study networking from the top down, 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

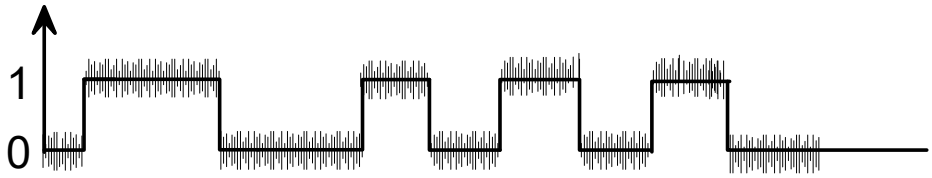
1 Byte per second = 1 B/s = 8 b/s

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

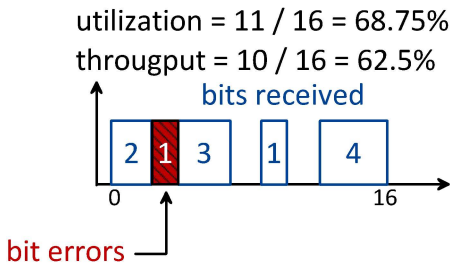
Capacity (bandwidth) is the maximum feasible data rate on a transmission medium. It is determined by the characteristics of the transmitter, the medium, and the receiver. The fundamental limits are the speed of the physical circuits and the **signal to noise ratio (SNR)** at the receiver. The receiver 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.



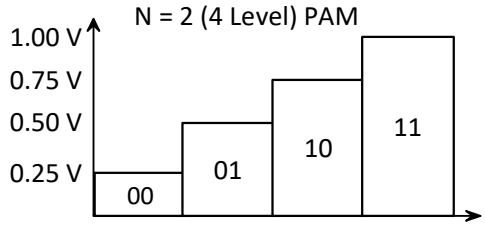
In the figure, the capacity in a certain transmission time is 16 bits — 11 bits are sent in 3 groups, so the utilization is 11/16, and only 10 bits are correctly received giving a throughput of 10/16.

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

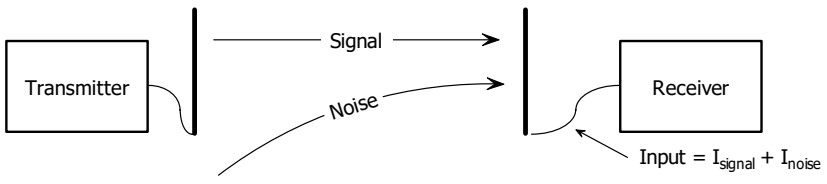
Data rate = (3300 symbols / second) $\times$ (10 bits / symbol) = 33,000 bps

Slides 7 – 14 Modeling Information — Signals and Noise

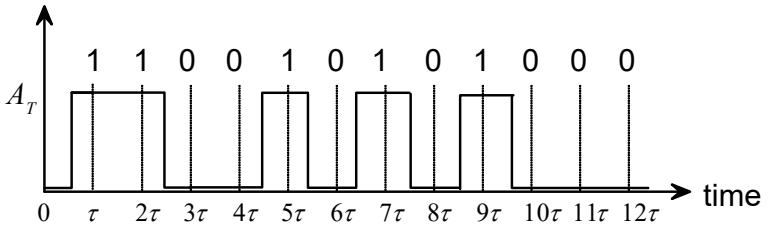
The word **information** is used to refer to a transmission that improves the likelihood of estimating the outcome of a test (the answers to a question). The universe of possible outcomes may be a finite set (yes/no, day of week, 256 pixel colors, etc) or an infinite set ("you won't believe what happened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

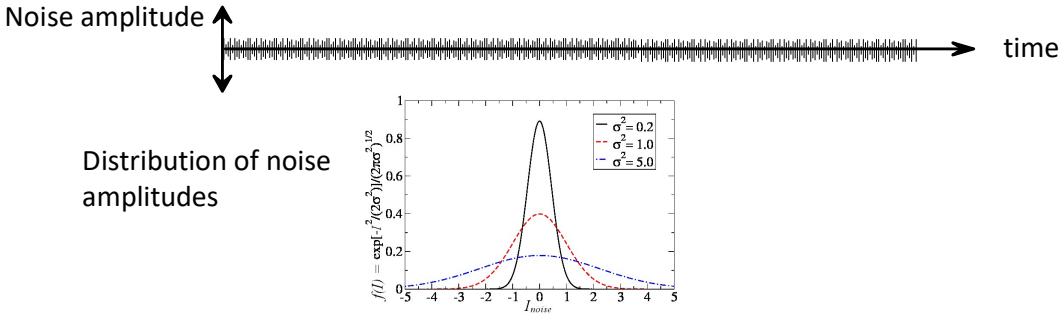
Sources of noise include interference from similar equipment, rounding-off errors, basic resolution of the detector, etc. Communication does not determine a unique outcome, but only an estimate subject to a probability of successful detection.



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 $A < A_T$.



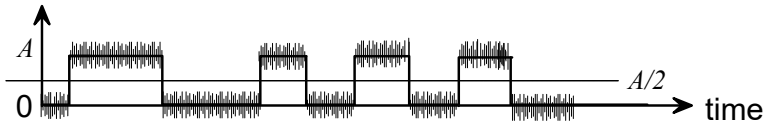
The noise added to the system this system can be characterized as "white noise" — a random current whose amplitude (level) obeys a Gaussian distribution ("bell curve").



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

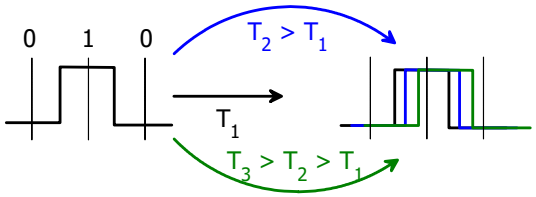
$$\text{Capacity} = W \times \log_2[1+SNR]$$

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.



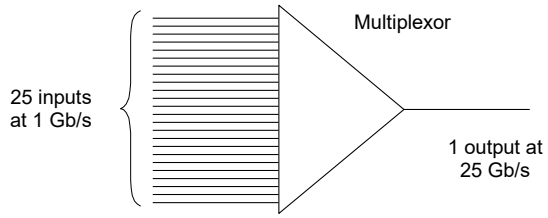
The receiver sees **inter-symbol interference (ISI)** — the interference between one signal arriving more than once at different times. If the ISI is not too bad, the receiver calculates the optimum time to sample the signals — at the time that they all agree.

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.

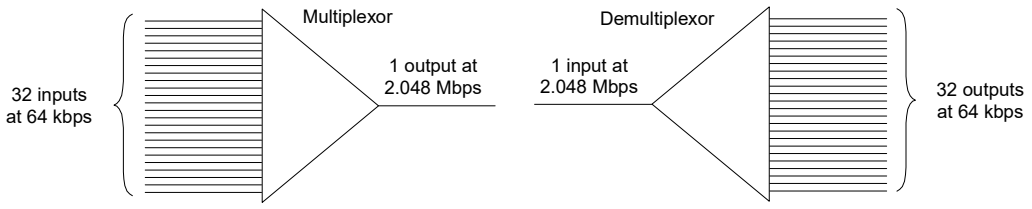


Combining data streams into one aggregate data streams is called **multiplexing**. Two basic methods of multiplexing are **Frequency Division Multiplexing (FDM)** and **Time Division Multiplexing (TDM)**.

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.

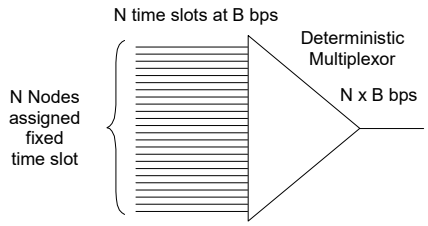
The **GSM** system for 2G and 3G cellular telephone systems uses a form of mixed multiplexing, combining FDM and TDM. The GSM radio frequency band of 25 MHz is divides into 125 channels with a bandwidth of 200 kHz per channel that is used to transmit data at a rate of 270 kbps. The 270 kbps data stream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

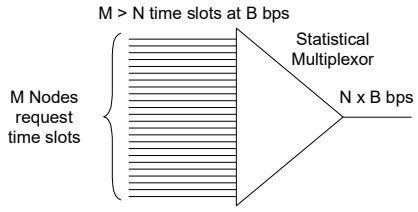
$P(k,T,\lambda)$ = probability of k bits arriving during T seconds when the average number of bits is λT

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

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

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

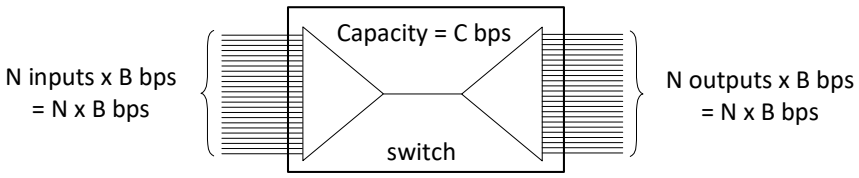
In order to lower the cost of high capacity systems, VBR data streams can be multiplexed using **statistical multiplexing**. A system serving M nodes provides N time slots, where M>N. The average data rate λ is less than the peak data rate B and so the total capacity = N x B can be set to a value larger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



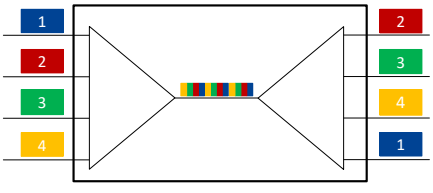
In this example

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.

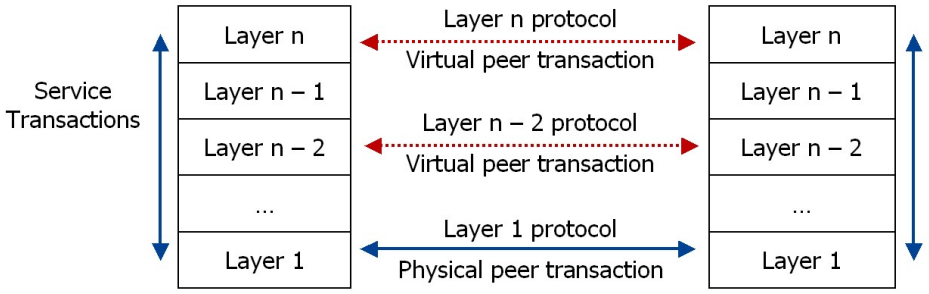
The division of activities in the communication hardware between transmission and management led to the concept of **layered protocols**. Transmission is defined at the **Physical Layer (PHY)** and management is defined at the **Data Link Layer** that supervises the PHY from above.

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

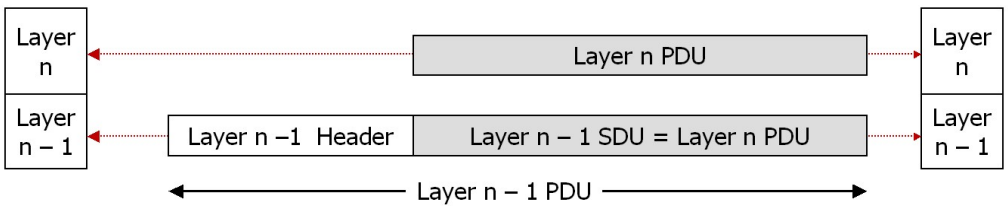


Protocol information at layer n is transmitted to the layer n peer in a **header** added to the message, in a process called **encapsulation**.

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>

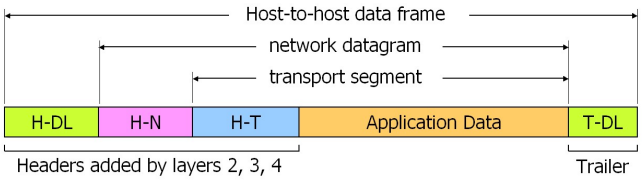
PDUs in the Internet protocols have specific names and address types:

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:

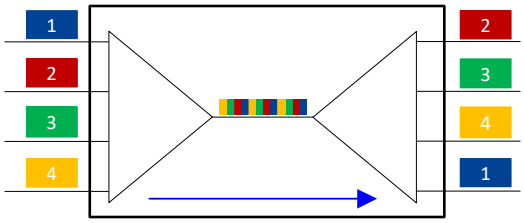
Preamble	**Start**	**Destination Address**	**Source Address**	**Type or Length**	**Data**	**CRC**
7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46 – 1500 bytes	4 bytes

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.

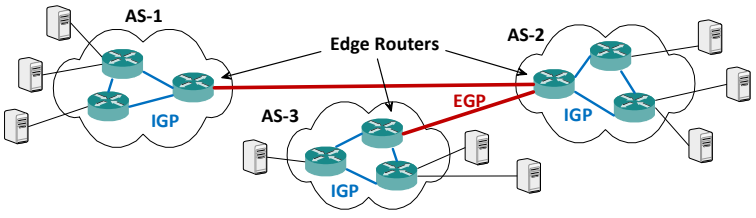
An equivalent system is the **subnet mask**, formed by writing 1 for every bit of the network number and 0 for every bit of the host number. The mask is used to allow hosts to identify other hosts on the same subnet: all hosts on the same subnet will give the same result when a bitwise AND operation 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.

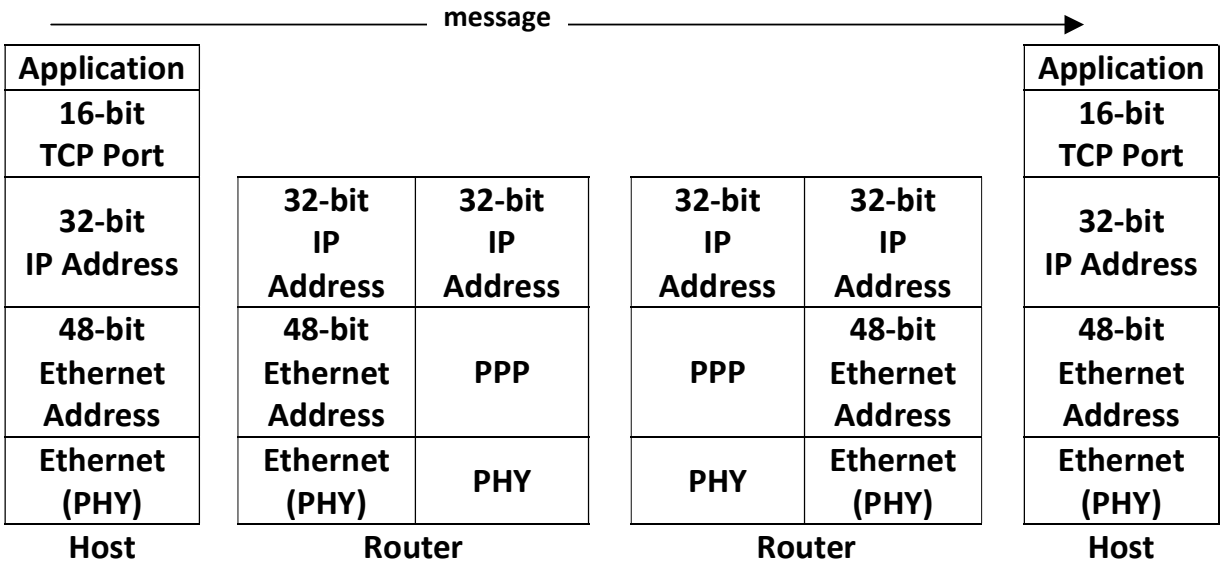
At the highest level, the Internet is modeled as a set of **Autonomous Systems (AS)**. An AS is the set of nodes managed by one organization.



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.