
EEE 545
COMPUTER NETWORKS 1

2.8.1-Ethernet Packet Format

- Here is the format of a typical Ethernet packet (DIX specification); it is still used for newer, faster Ethernets.

Ethernet Packet Format

- The destination and source addresses are 48-bit quantities; the type is 16 bits, the data length is variable up to a maximum of 1500 bytes, and the final CRC (cyclic redundancy check) checksum is 32 bits. The checksum is added by the Ethernet hardware, never by the host software. There is also a preamble, not shown: a block of 1 bits followed by a 0, in the front of the packet, for synchronization. The type field identifies the next higher protocol layer; a few common type values are 0x0800 = IP, 0x8137 = IPX, 0x0806 = ARP.

2.8.1 – Ethernet Packet Format

- Each Ethernet card has a (hopefully unique) physical address in ROM; by default any packet sent to this address will be received by the board and passed up to the host system. Packets addressed to other physical addresses will be seen by the card, but ignored (by default). All Ethernet devices also agree on a broadcast address of all 1's: a packet sent to the broadcast address will be delivered to all attached hosts.
- It is sometimes possible to change the physical address of a given card in software. It is almost universally possible to put a given card into promiscuous mode, meaning that all packets on the network, no matter what the destination address, are delivered to the attached host.

2.8.2–Time Slot and Collisions

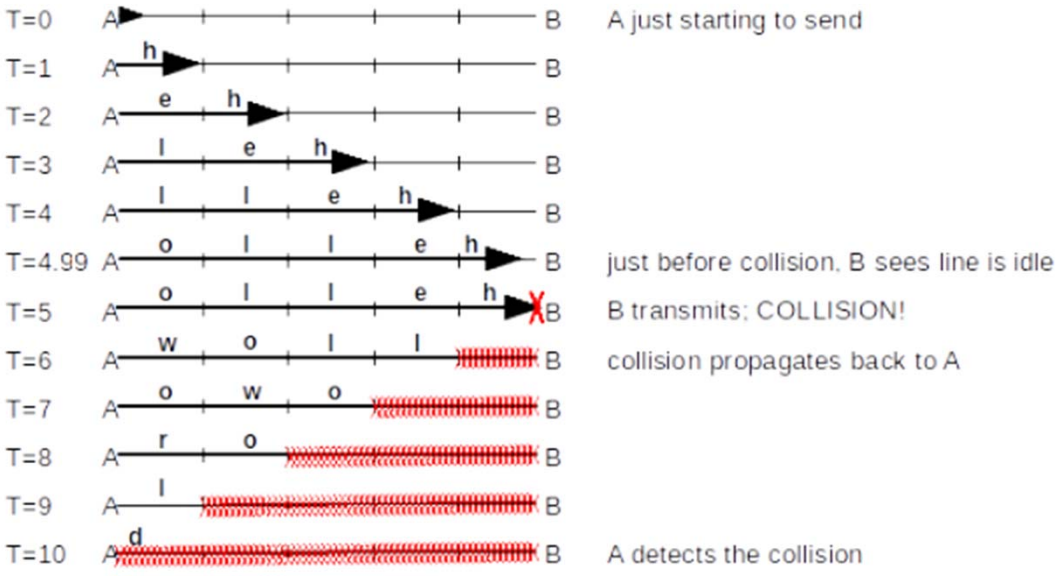
- The diameter of an Ethernet is the maximum distance between any pair of stations. The actual total length of cable can be much greater than this, if, for example, the topology is a “star” configuration. The maximum allowed diameter, measured in bits, is limited to 232 (a sample “budget” for this is below). This makes the round-trip-time 464 bits. As each station involved in a collision discovers it, it transmits a special jam signal of up to 48 bits. These 48 jam bits bring the total above to 512 bits, or 64 bytes. The time to send these 512 bits is the slot time of an Ethernet; time intervals on Ethernet are often described in bit times but in conventional time units the slot time is 51.2 μsec (for 10Mbps Ethernet system).
- The value of the slot time determines several subsequent aspects of Ethernet. If a station has transmitted for one slot time, then no collision can occur (unless there is a hardware error) for the remainder of that packet. This is because one slot time is enough time for any other station to have realized that the first station has started transmitting, so after that time they will wait for the first station to finish. Thus, after one slot time a station is said to have acquired the network. The slot time is also used as the basic interval for retransmission scheduling.

2.8.2–Time Slot and Collisions

- Conversely, a collision can be received, in principle, at any point up until the end of the slot time. As a result, Ethernet has a minimum packet size, equal to the slot time, i.e 64 bytes (or 46 bytes in the data portion). A station transmitting a packet this size is assured that if a collision were to occur, the sender would detect it (and be able to apply the retransmission algorithm). Smaller packets might collide and yet the sender not know it, ultimately leading to greatly reduced throughput.
- If we need to send less than 46 bytes of data (for example, a 40-byte TCP ACK packet), the Ethernet packet must be padded out to the minimum length. As a result, all protocols running on top of Ethernet need to provide some way to specify the actual data length, as it cannot be inferred from the received packet size.

2.8.2 Time Slot and Collisions

- As a specific example of a collision occurring as late as possible, consider the diagram below. A and B are 5 units apart, and the bandwidth is 1 byte/unit. A begins sending “hello world” at T=0; B starts sending just as A’s message arrives, at T=5. B has listened before transmitting, but A’s signal was not yet evident. A doesn’t discover the collision until 10 units have elapsed, which is twice the distance.



2.8.2–Time Slot and Collisions

- Implicit in the delay budget table above is the “length” of a bit. The speed of propagation in copper is about 0.77c, where c=3x10^8 m/sec = 300 m/μs is the speed of light in vacuum. So, in 0.1 microseconds (the time to send one bit at 10 Mbps), the signal propagates approximately 0.77xcx10^-7 = 23 meters.
- Ethernet packets also have a maximum packet size, of 1500 bytes. This limit is primarily for the sake of fairness, so one station cannot unduly monopolize the cable (and also so stations can reserve buffers guaranteed to hold an entire packet). At one time hardware vendors often marketed their own incompatible “extensions” to Ethernet which enlarged the maximum packet size to as much as 4KB. There is no technical reason, actually, not to do this, except compatibility.

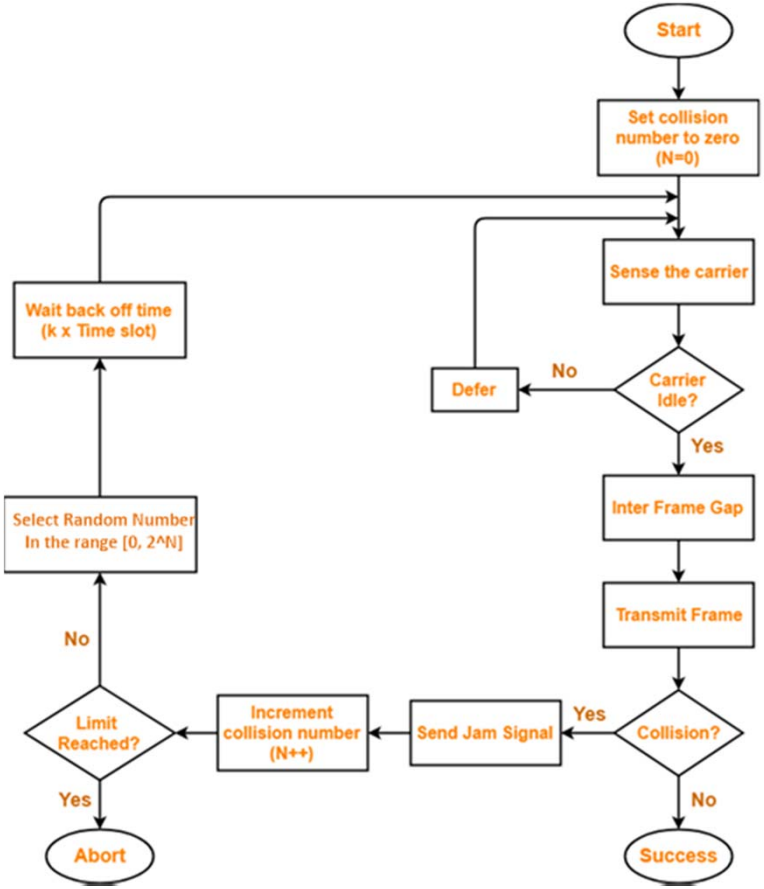
2.8.3–Exponential Backoff Algorithm

- Whenever there is a collision the exponential backoff algorithm – operating at the MAC layer – is used to determine when each station will retry its transmission. Backoff here is called exponential because the range from which the backoff value is chosen is doubled after every successive collision involving the same packet. Here is the full Ethernet transmission algorithm, including backoff and retransmissions:
- 1. Listen before transmitting (“carrier detect”)
- 2. If line is busy, wait for sender to stop and then wait an additional 9.6 microseconds (96 bits). One consequence of this is that there is always a 96-bit(double of jam bits) gap between packets, so packets do not run together.
- 3. Transmit while simultaneously monitoring for collisions
- 4. If a collision does occur, send the jam signal, and choose a backoff time as follows: For transmission N, $1 \leq N \leq 10$ (N=0 represents the original attempt), choose k randomly with $0 \leq k < 2^N$. Wait k slot times (kx51.2 μsec, minimum size of ethernet frame = 64 bytes = 512 bits. Time taken to transmit a minimum size ethernet frame on a 10Mbits link = 512/10,000,000 = 51.2 microseconds.). Then check if the line is idle, waiting if necessary for someone else to finish, and then retry step 3. For $11 \leq N \leq 15$, choose k randomly with $0 \leq k < 1024$ (= 210)
- 5. If we reach N=16 (16 transmission attempts), give up.

2.8.3–Exponential Backoff Algorithm

- If an Ethernet sender does not reach step 5, there is a very high probability that the packet was delivered successfully. Exponential backoff means that if two hosts have waited for a third to finish and transmit simultaneously, and collide, then when N=1 they have a 50% chance of recollision; when N=2 there is a 25% chance, etc.
- When $N \geq 10$ the maximum wait is 52 milliseconds; without this cutoff the maximum wait at N=15 would be 1.5 seconds. As indicated above in the minimum-packet-size discussion, this retransmission strategy assumes that the sender is able to detect the collision while it is still sending, so it knows that the packet must be resent.

2.8.3-Exponential Backoff Algorithm



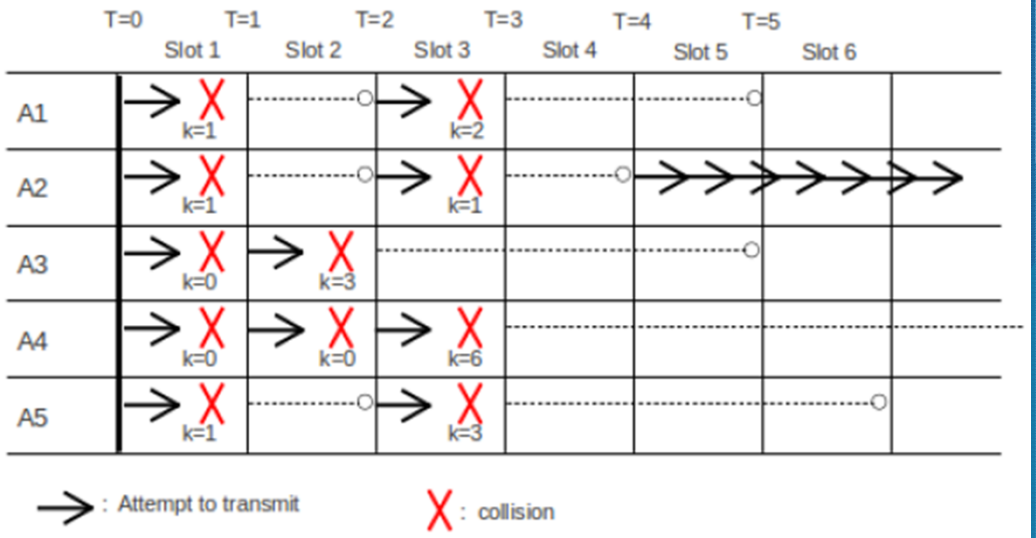
Exponential Backoff Algorithm

2.8.3–Exponential Backoff Algorithm

- In the following diagram is an example of several stations attempting to transmit all at once, and using the above transmission/backoff algorithm to sort out who actually gets to acquire the channel.
- We assume we have five prospective senders A1, A2, A3, A4 and A5, all waiting for a sixth station to finish. We will assume that collision detection always takes one slot time (it will take much less for nodes closer together) and that the slot start-times for each station are synchronized; this allows us to measure time in slots. A solid arrow at the start of a slot means that sender began transmission in that slot; a red X signifies a collision. If a collision occurs, the backoff value k is shown underneath. A dashed line shows the station waiting k slots for its next attempt.

2.8.3–Exponential Backoff Algorithm

- At T=0 we assume the transmitting station finishes, and all the Ai transmit and collide. At T=1, then, each of the Ai has discovered the collision; each chooses a random k<2. Let us assume that A1 chooses k=1, A2 chooses k=1, A3 chooses k=0, A4 chooses k=0, and A5 chooses k=1.
- Those stations choosing k=0 will retransmit immediately, at T=1. This means A3 and A4 collide again, and at T=2 they now choose random k<4. We will Assume A3 chooses k=3 and A4 chooses k=0; A3 will try again at T=2+3=5 while A4 will try again at T=2, that is, now.
- At T=2, we now have the original A1, A2, and A5 transmitting for the second time, while A4 trying again for the third time. They collide. Let us suppose A1 chooses k=2, A2 chooses k=1, A5 chooses k=3, and A4 chooses k=6 (A4 is choosing k<8 at random). Their scheduled transmission attempt times are now A1 at T=3+2=5, A2 at T=4, A5 at T=6, and A4 at T=9.
- At T=3, nobody attempts to transmit. But at T=4, A2 is the only station to transmit, and so successfully seizes the channel. By the time T=5 rolls around, A1 and A3 will check the channel, that is, listen first, and wait for A2 to finish. At T=9, A4 will check the channel again, and also begin waiting for A2 to finish.

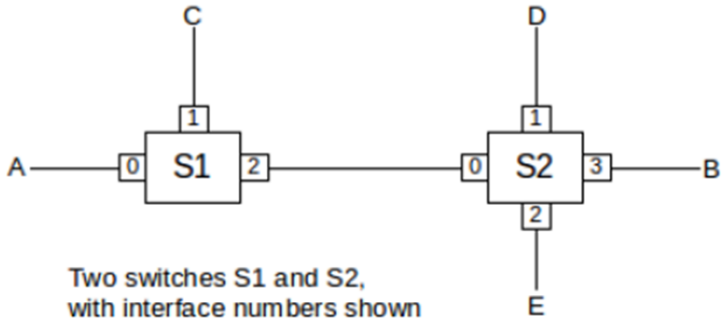


2.9 Ethernet Switches

- Switches join separate physical Ethernets (or sometimes Ethernets and other kinds of networks). A switch has two or more Ethernet interfaces; when a packet is received on one interface it is retransmitted on one or more other interfaces. Only valid packets are forwarded; collisions (possible) are not propagated. The term collision domain is sometimes used to describe the region of an Ethernet in between switches; a given collision propagates only within its collision domain. All the collision-detection rules, including the rules for maximum network diameter, apply only to collision domains, and not to the larger “virtual Ethernets” created by stringing collision domains together with switches.

2.9.1 Datagram Forwarding

- In the datagram-forwarding model of packet delivery, packet headers contain a destination address. It is up to the intervening switches or routers to look at this address and get the packet to the correct destination.
- In the diagram beside, switch S1 has interfaces 0, 1 and 2, and S2 has interfaces 0,1,2,3. If A is to send a packet P to B, S1 must know that P must be forwarded out interface 2 and S2 must know P must be forwarded out interface 3. In datagram forwarding this is achieved by providing each switch with a forwarding table of xdestination,next_hopy pairs. When a packet arrives, the switch looks up the destination address (presumed globally unique) in this table, and finds the next_hop information: the immediate-neighbor address to which – or interface by which-the packet should be forwarded in order to bring it one step closer to its final destination.



Two switches S1 and S2, with interface numbers shown

S1	
destination	next_hop
A	0
C	1
B	2
D	2
E	2

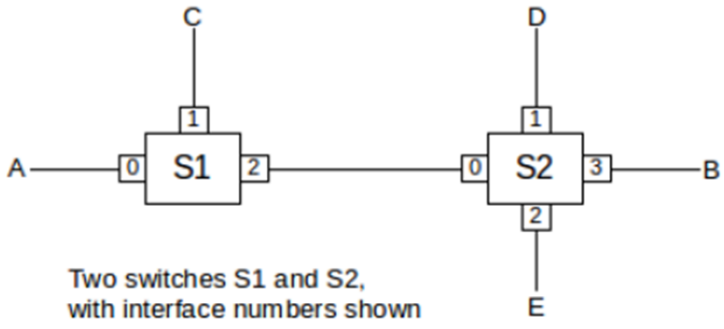
Table for S1

S2	
destination	next_hop
A,C	0
D	1
E	2
B	3

Table for S2

2.9.1 Datagram Forwarding

- For human readers, using neighbors in the next_hop column is usually much more readable. S1's table can now be written as follows (with consolidation of the entries for B, D and E)
- By convention, switching devices acting at the LAN layer and forwarding packets based on the LAN address are called switches (or, in earlier days, bridges), while such devices acting at the IP layer and forwarding on the IP address are called routers. Datagram forwarding is used both by Ethernet switches and by IP routers, though the destinations in Ethernet forwarding tables are individual nodes while the destinations in IP routers are entire networks (that is, sets of nodes).



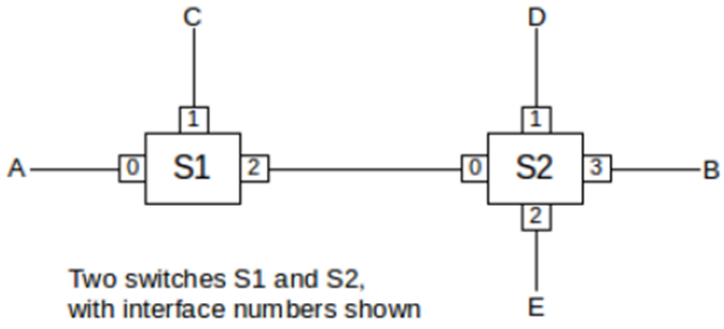
Two switches S1 and S2, with interface numbers shown

S1	
destination	next_hop
A	A
C	C
B,D,E	S2

Table for S1

2.9.1 Datagram Forwarding

- In IP routers within end-user sites it is common for a forwarding table to include a catchall default entry, matching any IP address that is nonlocal and so needs to be routed out into the Internet at large. Unlike the consolidated entries for B, D and E in the table above for S1, which likely would have to be implemented as actual separate entries, a default entry is a single record representing where to forward the packet if no other destination match is found. Here is a forwarding table for S1, above, with a default entry replacing the last three entries:



Two switches S1 and S2, with interface numbers shown

S1	
destination	next_hop
A	0
C	1
default	2

Table for S1 forwarding with IP Router

2.10 More about Packages

- When a router or switch receives a packet, it (generally) reads in the entire packet before looking at the header to decide to what next node to forward it. This is known as **store-and-forward**, and introduces a forwarding delay equal to the time needed to read in the entire packet. For individual packets this **forwarding delay** is hard to avoid (though some switches do implement **cut-through** switching to begin forwarding a packet before it has fully arrived), but if one is sending a long train of packets then by keeping multiple packets en route at the same time one can essentially eliminate the significance of the forwarding delay;

2.10 More about Packages

- **Bandwidth delay**, is sending 1000 Bytes at 20 Bytes/millisecond will take 50 ms. This is a per-link delay.
- **Propagation delay** due to the speed of light. For example, if you start sending a packet right now on a 5000-km cable across the US with a propagation speed of 200 m/μsec (= 200 km/ms, about 2/3 the speed of light in vacuum), the first bit will not arrive at the destination until 25 ms later. The bandwidth delay then determines how much after that the entire packet will take to arrive.
- **Store-and-forward(Transmission) delay,** equal to the sum of the bandwidth delays out of each router along the path
- **Queuing delay**, or waiting in line at busy routers. At bad moments this can exceed 1 sec, though that is rare. Generally it is less than 10 ms and often is less than 1 ms. Queuing delay is the only delay component amenable to reduction through careful engineering.
- **Processing Delay** depends on amount of data to process, software implementation, computer hardware, and other activities of computer.

  Often very small compared to transmission and propagation delay

Delay Example

Ex: R is an intermediate point (router, repeater etc.) between A and B. All the connections are made of copper whose transmission speed is 0.7c. If 100B of digital data wants to be transmitted by using 8Mb/s technology. What will be the total time for the data to arrive point B? (Unmentioned delays are assumed to be zero)

A----------→R----------→B

100km 100km

$T_{total} = 2 \times T_{BW} + 2 \times T_{prop}$

$T_{BW} = \frac{100 \times 8}{8 \times 10^6} = 100\mu s$ $T_{prop} = \frac{100 \times 10^3}{2{,}1 \times 10^8} = 476\mu s$

$T_{total} = 2 \times 100\mu s + 2 \times 476\mu s = \mathbf{1152\mu s}$

2.11 Ethernet Learning Algorithm

- Ethernet switches use datagram forwarding. The trick is to build their forwarding tables without any cooperation from ordinary, non-switch hosts. Switches start out with empty forwarding tables, and build them through a learning process. If a switch does not have an entry for a particular destination, it will fall back to broadcast: it will forward the packet out every interface other than the one on which the packet arrived. The availability of fallback-to-broadcast is what makes it possible for Ethernet switches to learn their forwarding tables without any switch-to-switch or switch-to-host communication or coordination.

2.11 Ethernet Learning Algorithm

- A switch learns address locations as follows: for each interface, the switch maintains a table of physical MAC (Media Access Control) addresses that have appeared as **source addresses** in packets arriving via that interface. The switch thus knows that to reach these addresses, if one of them later shows up as a **destination address**, the packet needs to be sent only via that interface. Specifically, when a packet arrives on interface I with source address S and destination unicast address D, the switch enters <S,I> into its forwarding table.

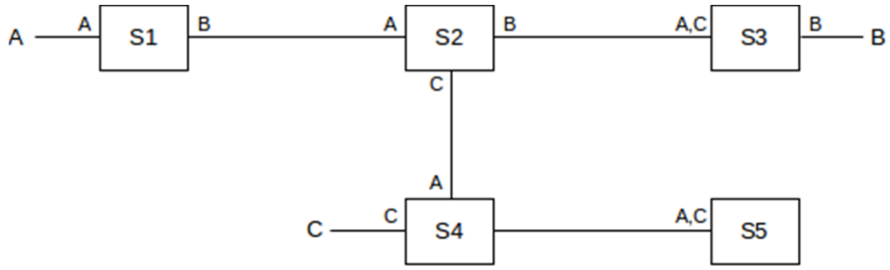
2.11 Ethernet Learning Algorithm

- To actually deliver the packet, the switch also looks up D in the forwarding table. If there is an entry <D,J> with J≠I that is, D is known to be reached via interface J – then the switch forwards the packet out interface J. If J=I, that is, the packet has arrived on the same interfaces by which the destination is reached, then the packet does not get forwarded at all; it presumably arrived at interface I only because that interface was connected to a shared Ethernet segment that also either contained D or contained another switch that would bring the packet closer to D. If there is no entry for D, the switch must forward the packet out all interfaces J with J ≠ I; this represents the fallback to broadcast. After a short while, this fallback to broadcast is needed less and less often, as switches learn where the active hosts are located. (However, in some switch implementations, forwarding tables also include timestamps, and entries are removed if they have not been used for, say, five minutes.)

2.11 Ethernet Learning Algorithm

- If the destination address D is the broadcast address, or, for many switches, a multicast address, broadcast is required. Some switches try to keep track of multicast groups, so as to forward multicast traffic only out interfaces with known subscribers;

- In the diagram above, each switch's tables are indicated by listing near each interface the destinations (identified by MAC addresses) known to be reachable by that interface. The entries shown are the result of the following packets:
- • A sends to B; all switches learn where A is
- • B sends to A; this packet goes directly to A; only S3, S2 and S1 learn where B is
- • C sends to B; S4 does not know where B is so this packet goes to S5; S2 does know where B is so the packet does not go to S1.



Five learning bridges after three packet transmissions