

## LAN simplifications

### Lecture 5

#### Multiple access

Many physical medium can be used for broadcasting: when any station send a message, all stations connected to the medium receives it.

We see examples of such media in practice, and how to efficiently use such medium to achieve communication.

#### References:

- CN: 6.1–6.2, 6.3.2, 6.7, 6.10, 6.11.1, 8.2.6, 8.2.7.
- RFCs: 826 (ARP), 951, 1542.

Network(0234B)

Network(0234B)-5.1

- **Local Area Networks** (LANs) are owned by a single entity, typically run no more than a few hundred meters.
- They are **physically secured**, so that all **users are trusted** to follow even conventions that would restrict their sending rates.
- This makes it practical to **have a medium shared by many users**, and let **everybody broadcast frames**.  
No party will "jam" the medium and leave everybody toasted.
- Such systems are **cheap to build**, as:
  1. **No extra computer is needed** to forward data among computers.
  2. **Simpler network software**: just broadcast to the shared medium.
  3. We don't worry about who to connect to whom. **Just connect all the wires.**

## Multiple Access mechanisms

We first deal with a more basic problem: **when** each station is allowed to use the medium? In other words, how to **schedule** the single medium?

1. We can do basically **no scheduling**. So everybody can use the medium **at any time**.

If a receiver receives two frames at the same time, it causes an error that must be dealt with somehow.

This is the approach taken by Ethernet (802.3), and also in the default mode of Wireless LAN (802.11).

2. We can ask **one computer to do the allocation** of bandwidth, e.g., using TDM or FDM.

When an access point is present, wireless LAN can operate like this.

We will look into each of the protocols used.

Network(0234B)-5.2

Network(0234B)-5.3

## Pure Aloha

Ethernet uses a scheme that is modified from **Aloha**, a radio system previously used by Hawaiian universities.

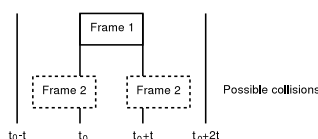
1. When one has a frame to send, send it down the broadcast media **as if it is the sole user**.
2. If a receiver **receives two frames at the same time**, a **collision** is said to happen, and **nothing is received**.  
Frames must be short, otherwise it gets too likely to be screwed up.
3. The higher layer notes the absence of ACK, and wait for a **random amount of time** (the **backoff** time) before resending the frame.  
The "randomization" is **vital**: otherwise there will be another collision when both stations resend their data.

Seems chaotic. But under the correct situations, performance is good!

## Condition for a collision

To simplify the discussion, assume **all frames are of length  $t$** .

Suppose a frame F1 is received beginning at  $t_0$ . Will another frame F2, start being received at time  $t_1$ , collide with it?



So there is a collision if  $t_0 - t < t_1 < t_0 + t$ .

We call the period of time  $(t_0 - t, t_0 + t)$  to be the **vulnerable** period: if another frame is received during this period, a collision occur. If there is no other frame sent during the vulnerable period, the transmission succeed.

Network(0234B)-5.4

Network(0234B)-5.5

## Performance of pure Aloha

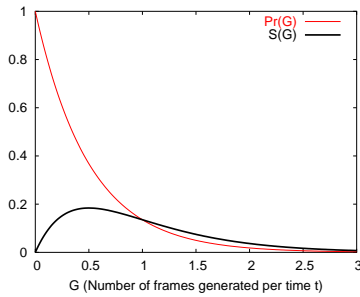
- Let  $G$  be the average number of frames that are received (with or without error) by a station during time  $t$  (i.e., generation rate is  $G/t$ ).
- Assume that frame generation is **random** and **time independent**. The number of frames  $k$  received in a duration of length  $\tau$  thus following a **Poisson distribution**:

$$\Pr[k] = \frac{(r\tau)^k e^{-r\tau}}{k!}$$

- We need a clean period (i.e.,  $k = 0$  frame generated) of  $\tau = 2t$ , so probability of success is  $e^{-2G}$ .
- Thus the rate of frame received per period  $t$  is  $S = Ge^{-2G}$ . This should be the same as the **rate of frames being sent** by all computers.

### Maximum medium utilization

Let's plot  $S$  against  $G$  to see when  $S$  is maximized:



So maximum performance happens when  $G = 0.5$ , i.e., 1 frame generated per  $2t$ , the length of vulnerable period. Maximum utilization is 18.4%.

Network(0234B)-5.6

### CSMA

In (early) Ethernet, the **transmission delay is much less than the frame time**. Then there is a simple way to do better...

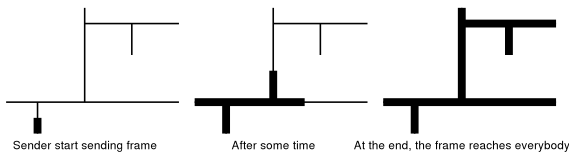
- **Idea:** what the receiver see is similar to what I'm seeing.
- So if I'm seeing a frame passing the wire now, there is **no reason to start a transmission**. We call this **carrier sensing**: before starting a transmission, **sense** the channel, and transmit only if it is now clear.
- What to do if it is not clear? The sender might **monitor** the channel to wait until it becomes clear, and **send** at that time.
- But... if **more than one sender** is doing this simultaneously, it causes a collision. So its better to instead **backoff**, i.e., after finding the channel is clear, wait for a random amount of time before retry.

The scheme is called **Carrier Sense Multiple Access**, or CSMA in short.

Network(0234B)-5.7

### Channel seizing

What happen when a sender sends a frame, in presence of delay?



So after an amount of time  $t_D$  when the frame travels a distance **network diameter  $D$** , everybody can sense a conversation is in progress, and thus **will not interfere**. We say the sender **seized the channel**.

If  $t_D$  is large with respect to  $t$  (the time needed to send a frame), then carrier sensing is not useful. So every CSMA system **limits  $D$  to be small**.

Network(0234B)-5.8

### CSMA-CD

One final modification to Aloha bring us to the Ethernet scheme:

- When transmission is in progress, the sender **monitors the voltage** on the wire to see whether it is the same as what is being sent.
- If it differs by too much, it is because of **another sender**: **collision** had occurred, so the current transmission is not useful.
- This is called **Collision Detection**, and the scheme is called **CSMA-CD** (Carrier Sensing Multiple Access with Collision Detection).
- We have to **continue to send** for  $t_D$  to make sure everybody knows about the collision. Then it stops.
- After sending for  $t_D$  without collision, the channel is seized. But the **sender don't know about it yet**, since a collision can need a further  $t_D$  to propagate back. After sending for  $2t_D$ , sender knows channel is seized.

Network(0234B)-5.9

### Minimum length

The Ethernet scheme makes sure that every collision is detected, by **always send at least for  $2t_D$ , even if the frame is shorter**.

- When CSMA-CD is used, the frame length is at least **64 bytes = 512 bits** (discounting preamble). I.e., Network packet + pad occupies at least 46 bytes (this is the only purpose of the pad field).
- For **10Mbps** Ethernet, the shortest frame is 64x8 bit:  $t = 512 \times 10^{-7}$ s.
- To have  $t \geq 2t_D$ ,  $t_D$  is at most  $256 \times 10^{-7}$ s.
- Electrons travel at  $2.3 \times 10^8$  m/s in copper. So this means 5.88km.
- To **allow for 4 repeaters** and the processing delay, 10Mbps Ethernet requires the distance between two hosts to be at most 2.5km.  
Repeaters are 2-port devices to amplify signals. Hubs = Multi-port repeater.

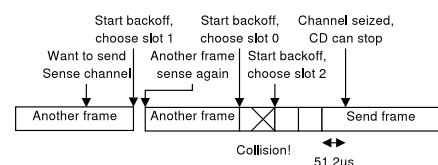
This distance is **proportionally smaller if the network speeds up**: 100Mbps network would have distance at 250m.

Network(0234B)-5.10

### Contention slots

The  $512 \times 10^{-7}$ s, i.e., 51.2  $\mu$ s is an important parameter, in the **design of the actual backoff algorithm** for Ethernet.

- We don't really need a truly "random" time: what makes a difference is just whether two stations choose times that are  $< 51.2 \mu$ s apart.
- Each computer sense the channel, and the time after the signal (for frame or collision) stops is divided into 51.2  $\mu$ s **contention slots**.
- So the actual backoff looks like this:



Network(0234B)-5.11

### The Truncated binary exponential backoff algorithm

The actual algorithm used: the slot numbers chosen is a **random number** among **exponentially large space**.

- For the first try, choose a slot number randomly among {0, 1}.
- If a collision is found, choose a random number again, this time from the numbers {0, 1, 2, 3}. If it still collide, choose among {0, 1, 2, 3, 4, 5, 6, 7}, etc. **Every collision** causes the random number space to **double**.
- If a frame is **sent successfully** by some other station, **restart**.
- If it fails more than 9 times, don't further double the space. Instead try choosing a random number among {0, ..., 1023}. At the 16-th collision, stop and report failure.

**Advantage:** if few channels contend, a slot is found in a few tries. If many channels contend for a slot, soon we will be trying random numbers which are large, and at that time some station is likely to succeed.

Network(0234B)-5.12

### Analysis of backoff using slots

Suppose  $n$  stations want to send, each choosing a slot number randomly in a range of  $m$  numbers.

- Then each station has a probability of  $1/m$  to choose the next slot.
- Probability that the next slot is idle =  $(1 - 1/m)^n$ .
- Probability that the next slot is used successfully =  $\frac{n}{m}(1 - 1/m)^{n-1}$
- Remaining: the next slot is wasted due to collision.

Best  $m$  is  $n$ , when the probability that a frame is sent successfully in the next slot is  $(1 - 1/n)^{n-1}$ . This approaches  $1/e$  as  $n$  increases.

In the Ethernet scheme, we expect to waste  $\log_2 n$  slots on collisions (so that  $m \approx n$ ), and  $e$  slots before a successful slot, so have to wait for  $\log_2 n + e$  slots on average.

Network(0234B)-5.13

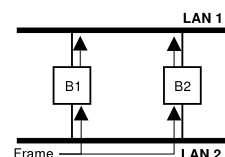
### Switched Ethernet

- When there are too many users, a **single shared medium has insufficient bandwidth**. We want to use **more than one LAN**.
- How computers on different LANs communicate? One might use a **router** to connect to both media, forward packets **in the network layer**.
- A simpler solution: connect LANs using a hardware **bridge**.  
Or a **switch**, i.e., a bridge with more than 2 ports.
- Bridges **remembers hardware addresses** of computers as frames come across. After that, frames towards that computer won't be forwarded to incorrect LAN.  
Before a computer send its 1st frame, frames to it are always forwarded.
- Bridges and switches **keep frames in buffers**: they **wait** until the other LAN is idle, rather than colliding. So we say different LANs have different **collision domains**.

Network(0234B)-5.14

### Bridge redundancy and loops

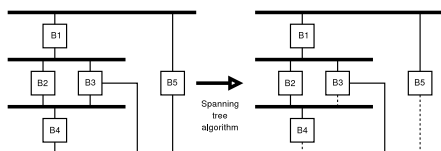
- Sometimes we want **multiple bridges** (or switches, we use the words interchangeably) to connect LANs: **if one fail the other takes over**.
- But... multiple bridges cause **duplicated frames** to be forwarded, which collides, get resent, and reach the other bridge, forwarded again, ...



- Need two things: (1) an algorithm to **shut down** some of the bridge ports so that a **spanning tree** remain, and (2) a way for the these ports to be **brought up** again when a bridge fail.  
A spanning tree keep just enough edges to keep the graph connected.

Network(0234B)-5.15

### Illustration of spanning tree



#### Which spanning tree?

1. Each bridge has a "bridge ID" (BID), the least BID bridge is the **root**.
2. The spanning tree that will be used is the one to **connect each LAN and switch to the root with the least "cost"**.  
Each bridge knows what is the "cost"  $\Delta c$  to reach the bridge from each port. E.g., the cost for 10Mbps network might be larger than the cost for a 100Mbps one.

The bridges follows the **distributed spanning tree algorithm** to send **configuration bridge PDU** ("cBPDU") to compute the tree.

Network(0234B)-5.16

### Distributed Spanning Tree algorithm

- The **root** periodically **generates cBPDU** (root,0,root). Initially every bridge believe itself to be root and thus send such frames.
- When bridge B receives cBPDU (r,c,b): if  $r >= B$ , it is ignored. Otherwise it **knows itself is not root**, and **stop generating cBPDU**.
- Each bridge stores  $\min_r$ , the minimum r received. cBPDUs with  $r > \min_r$  are ignored. It knows the cost to  $\min_r$  is at most  $\min_c = c + \Delta c$ , which is also stored. It may **forward** the cBPDU by sending ( $\min_r, \min_c, B$ ).
- In each LAN, only its **designated bridge** should forward cBPDU. The designated bridge is the least ID bridge providing the least cost path to the LAN. When a bridge **updates  $\min_r$  or  $\min_c$** , it believes itself to be **designated for all other ports**.
- Once bridge B receives cBPDU (r,c,b) from a LAN with  $r = \min_r$  but  $c < \min_c$ , or  $c = \min_c$  and  $b < B$ , it knows it is **not the designated bridge for that LAN**, so it **stops forwarding** in that LAN.

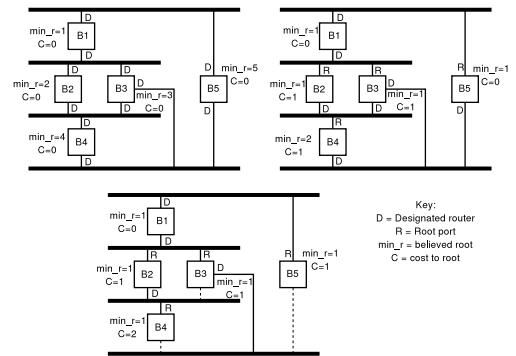
Network(0234B)-5.17

### The spanning tree, recalculation

- At the end **only one bridge of the network** claims to be **root**, and **only one bridge of each LAN** claims to be designated.
- The **spanning tree is realized** by each bridge enabling only the ports to LANs that it is a designated bridge, plus one more port (the **root port**) from where it gets the least cost cBPDU.
- The **root** will continue to **generate** cBPDUs, and **designated** bridges will continue to forward cBPDUs, even after the spanning tree is found.
- Each bridge **keeps a timer** which fires off if it **doesn't receive** the generated or forwarded frame **for too long**, at that point the bridge will start from scratch (claiming itself as root).
- This happen if the **root** or the **designated bridge of a LAN dies**. The algorithm will stabilize on the spanning tree of the new network topology.

Network(0234B)-5.18

### Distributed Spanning Tree: Example computation



Network(0234B)-5.19

### Gigabit Ethernet: only switched

- 100Mbps "Fast Ethernet" users use switches more often, especially because they becomes less expensive.  
It is virtually impossible to find 100Mbps hubs in the market now.
- When Gigabit Ethernet is designed, the **network diameter** falls too small (25m) to be useful.
- The answer: forget about repeaters, hubs and multiple access.
- Gigabit and 10 Gigabit Ethernet works with **switches**, which route frames in the networks containing **point-to-point** links.
- CSMA-CD is not used**. Maximum length is determined by signal strength rather than protocol issues.

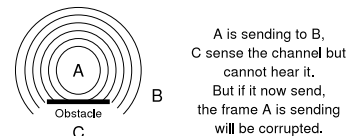
Network(0234B)-5.20

### Wireless complications

But there is still the **wireless LAN** where CSMA is used.

There is a complication, though: **not everybody can hear from everybody** (due to range problems and obstacles)!

This cause the **hidden station problem**, making CSMA ineffective:



Radio can work around obstacles, but the signal will be weakened.

Another problem: **collision detection no longer work**.

When sending in omnidirectional medium, your own strong signal at distance 0 overwhelm all other incoming signals!

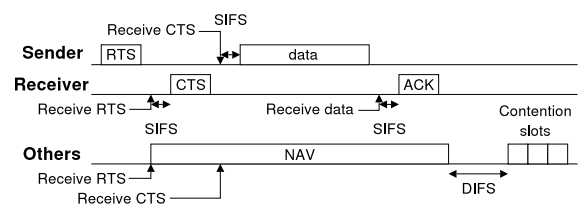
Network(0234B)-5.21

### Virtual channel sensing (Collision Avoidance)

- A collision occurs if another frame reaches the **receiver** at the time it receives. So we should try **detecting the receiver!** But... how?
- In the normal mode of wireless LAN (802.11), **virtual channel sensing** is used: sender sends a short frame called **Request to send (RTS)**, then receiver sends a short frames called **Clear to send (CTS)**. Sender then sends the actual data frame, and receiver sends an ACK.
- RTS and CTS contain the length of the frame to be sent, so other stations **avoid collision** by not sending during that period.
- A normal station seeing any communication will wait for a time called DIFS before backoff, which is **longer** than the time between the SIFS that separate RTS, CTS, data and ACK.  
DIFS=DCF Inter-Frame Space, SIFS=Short Inter-Frame Space.
- Collisions of data rarely happen. But RTS and CTS both can collide, and at such time the sender resend RTS after in a slot after **back-off**.

Network(0234B)-5.22

### Example 802.11 contention



Stations around have 3 chances to stop: the sender's RTS, the receiver's CTS, or the sender's data. Hopefully one gets through.

They allocate a **Network Allocation Vector (NAV)**, i.e., allocate the network assuming somebody is using it.

The propagation time is largely exaggerated: it is expected to be within 1 μs, where the SIFS is 18μs. In the next figure we will not draw it at all.

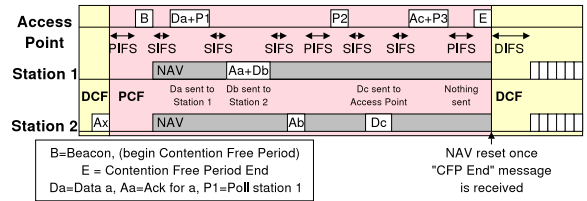
Network(0234B)-5.23

### PCF mode

- What we have described is **distributed coordination function (DCF)**. But when 802.11 is designed, there is desire to have a way to get **guaranteed** bandwidth for certain stations, which CSMA can't provide.
- If there is an **access point (AP)**, i.e., central station **everybody can hear** that coordinate the sharing of the channel, then there is a solution. 802.11 call this **point Coordination Function (PCF)**.
- The idea is, after a station **register** for PCF function, the AP will from time to time **poll** the station telling it to send a frame. No contention needed.
- But does the AP need contention to poll? No: PCF send frames just a **PIFS** (PCF Inter-Frame Space) after the end of last frame, which is between SIFS and DIFS. So it seize the channel before any contention.
- The AP do the polling periodically: **allocate a NAV** for a long period of time, poll each station, and cancel the NAV to restart DCF.

Network(0234B)-5.24

### Example



The AP announces the start of a **contention free (CF)** period, by sending a **beacon** before any station can contend.

Then the AP **poll each of the registered stations** in turn, those stations which have a frame to send will send it at the time it is allocated.

At the end, once each station is polled, the AP announce the CF period ends, so contention can start again.

Network(0234B)-5.25

### Other services of 802.11

- **Fragmentation:** When a frame passes through a 802.11 network, it may be fragmented, and reassembled at the AP. This is needed if the physical channel becomes too noisy: shorter frames are **easier to get through without error**.
- **Mobility:** With a **distribution service (DS)**, a station can physically move, and when it find that another station is closer, it can **handed-over** be to the new AP without getting a different network address.
- **Security:** With **Wired Equivalent Privacy (WEP)**, Wireless frames can be **encrypted** using a 56-bit key. Unluckily, various technical problem makes this easy to sidestep.

Network(0234B)-5.26

### Point-to-point communication over broadcast medium

How to achieve **point-to-point** communication, where there is exactly **one receiver** interested in what we send?

- Simple: Every packet has an **address (IP address)**, so **everybody listen to the medium and extract all packets**. Each computer then inspect the IP address, and **drop** the packet if it is **not for this computer**.
- Problem: this **uses a lot of CPU cycles of the computer**. If the datalink layer **hardware** (i.e., "LAN cards") can help us, it is preferable to **let the datalink layer do the filtering**.

But since the **datalink layer hardware has no knowledge of the network layer**, it cannot do the filter ingbased on IP addresses.

- Solution: Every LAN card has a **hardware address**. Each frame contains a **destination hardware address**: if it is not the hardware address of this LAN card, the frame is not forwarded to network layer. So we need to know the receiver's hardware address to send frames.

Network(0234B)-5.27

### Ethernet addressing scheme

- The Ethernet addressing scheme caters for **three types** of frames:
  1. **Unicast:** only one receiver is interested.
  2. **Multicast:** a group of receivers are interested.
  3. **Local broadcast:** everybody of the same LAN (or another LAN connected through bridges) is interested. This is a special case of multicast.
- Unicast addresses start with 0, multicast addresses start with 1. The broadcast address is 48-bits of 1's.
- Each LAN card (Ethernet or Wireless) has a 48-bit **hardware address** which is a **unicast address**. Each manufacturer is allocated a group of addresses, and they are **never reused**. In other words, hardware addresses are **globally unique**.

Network(0234B)-5.28

### Relating with IP addresses

Network layer (IP) **won't use hardware address**, since they provide no information for routing. Instead 32-bit IP addresses are used.

How IP addresses are related to hardware addresses?

- **Unicast** addresses: **not related at all!** In order to send a packet over Ethernet, the network layer has to **find the hardware address** of the destination computer, by broadcasting a question "who has the IP address aaa.bbb.ccc.ddd?", using the **Address Resolution Protocol (ARP)**. The destination replies the broadcast. In IPv6, the hardware address is usually part of the IPv6 address.
- **Multicast** addresses: each IP multicast address maps to an Ethernet multicast address. But the mapping is **not unique**: each Ethernet multicast address is used by 32 different IP multicast addresses. So datalink layer filter ingis incomplete. Reason: IP has 28 bits for specifying multicast addresses, but can only secure 23 bits in the Ethernet address space. The situation is worse in IPv6.

Network(0234B)-5.29