Lecture 14: Carrier Sense

Anirudh Sivaraman

2018/11/04

Last lecture, we looked at the ALOHA protocol for medium access control (MAC). The ALOHA protocol pioneered two main ideas: the ability to detect and handle collisions on a shared medium and the use of randomization to decide whether to transmit a packet or not. However, ALOHA was designed for situations where none of the users could hear any of the other users transmitting. Real-world wireless deployments are not always this bad, and oftentimes users *do* have the ability to hear each other's transmissions—an ability called carrier sense. This lecture, we'll look at a few examples of how to incorporate carrier sense into the MAC protocol.

1 Carrier sense

Carrier sense (or colloquially listen before you talk) is the ability for a user on a shared medium to know that some other user is transmitting data on that medium already. Typically, this is accomplished by having the user measure the current voltage levels on its wire or signal intensity on its receiver antenna and check if they exceed some threshold, which provides some evidence of ongoing transmissions in the medium. We won't worry about how the carrier sense ability itself is implemented, but instead look at how to use this ability in the design of MAC protocols.

2 Carrier sense in Ethernet

The first instantiations of Ethernet were as a shared medium, where multiple users tapped into a single shared line (or bus). To send packets to each other, these users would send packets into this shared line, which would then deliver packets to the appropriate receiver. Today's Ethernet deployments don't work this way anymore. Instead they use a switch, with multiple users sending packets into this switch using separate cables.

The difference between bus-based Ethernet and switch-based Ethernet is the difference between a shared and an unshared medium. In a switch-based Ethernet, user A can send a packet to user B and simultaneously user C can send a packet to user D—as long as the switch finds a way to connect up A to B and C to D. How the switch finds a way to connect A to B and C to D depends on whether the switch is input queued or output queued. Recall from lecture 12 that for an input queued switch, the switch needs to find a bipartite matching. For an output queued switch, this problem is much easier because each output can accommodate packets from all inputs in a particular hardware clock cycle.

Now, in a bus-based Ethernet only one pair of sender and receiver can commmunicate at a given instant because the bus is a shared communication medium that needs to be divided up across all senders and receivers; the bus can accommodate only one packet at a given instant. Throwing in multiple packets into the bus causes the packets to collide in the same way as ALOHA, i.e., the packets interfere with each other at the electrical voltage level to the point where the packets cannot be uniquely reconstructed anymore. Some recent research, e.g., ZigZag decoding [3], provides algorithms to recover two colliding packets using data from multiple packet collisions. However, I do not know of commercial link-layer technologies that support this capability.

Hence, bus-based Ethernet provides us a good (and early) example of a shared communication medium similar to ALOHA, but with the additional ability to detect transmissions of other users. We'll now briefly describe how the MAC protocol in bus-based Ethernet works.[1] This MAC protocol is called CSMA/CD for Carrier Sense Multiple Access with Collision Detection. It works as follows.

Before transmitting a packet, a user connected to an Ethernet bus senses if the medium is idle or busy transmitting some other user's packets. If the medium is busy, the user stays quiet and continues to sense the medium until it is idle. When the medium is idle, the user transmits the bits of the packet and then checks if each bit has collided with the transmission of another user. If so, the user backs off, by deciding not to try to transmit packets again for a randomly chosen duration of time. The user instead transmits a *jamming signal* on the bus that notifies all users on the Ethernet that there has been a collision that requires all of them to abort their transmissions. Each user that receives a jamming signal knows that anything that it receives just after the jamming signal has been received must be discarded because there was a collision on the medium that could have corrupted the data.

How is this random backoff time chosen? This time is picked from a uniform distribution between 1 and CW time units, where CW is called the contention window. CW doubles with every collision in a manner similar to exponential backoff until the packet is successfully received. The difference relative to ALOHA's doubling of probabilities is that the backoff time will always be a finite number within a particular range (1 to CW). So every user will eventually retransmit a packet, no matter how unlucky they are. By contrast, with ALOHA, there is always the possibility that a user gets unlucky on every transmission attempt because it decides whether to transmit or not in every attempt without any attention being paid to how long it has not transmitted so far.

3 Carrier sense in WiFi

Carrier sense is also used in WiFi. However, unlike Ethernet, a WiFi user cannot transmit and receive at the same time. To understand why this is the case, let's assume a simplified model of the WiFi physical layer where we have a separate transmit antenna to send out packets and a separate receive antenna to receive packets. At any given instant, the user's receive antenna receives EM waves from multiple different senders. In general, the closer a sender is to a receiver, the higher the intensity of the EM wave received by that receiver from that sender. If a user is transmitting, the user's own transmissions have a much higher EM wave intensity at the user's receiver relative to transmissions from any other user. This is because the user's transmit antenna is right next to the user's receive antenna (typically a few cm apart) and the other users are much further away. This means that the EM wave from the user's transmissions is going to completely overwhelm the EM wave from any other senders, making it to impossible to decode transmissions from any other senders while the user is transmitting. Some recent research (ca. 2013) that combines both circuit design and new algorithms shows that it is possible to build such *full-duplex* radios that can transmit and receive at the same time [2]. Some of this research is in the field trial phase for future wireless technologies [1].

Returning back to WiFi as it stands today, WiFi users typically cannot transmit and receive at the same time. Hence, the ability to detect collisions while the user is transmitting (which relies on the ability to receive and decode electrical signals when transmitting) is not an option for WiFi. However, a WiFi user can still detect ongoing transmissions for users that are close to it (carrier sense) before commencing any data transmission. Hence, in this respect, it is more capable than an ALOHA user. The MAC protocol for WiFi is called CSMA/CA for Carrier Sense Multiple Access with Collision Avoidance, because it tries to avoid collisions instead of detect them like Ethernet does.

The CSMA/CA algorithm works as follows. First, a WiFi user pick a random backoff interval I by sampling uniformly from the range 1 to CW time slots. It then waits for I idle slots, i.e., it decrements the backoff timer only during idle slots and not when carrier sense detects that the medium is busy. Once it has waited I idle slots, it transmits the packet entirely—instead of checking for a collision after every bit as Ethernet does. Note that such a check isn't even possible in WiFi because the user can't transmit and check for collisions (receive) simultaneously.

Once the user transmits the packet entirely, it then waits for an ACK from the receiver. If no ACK has been received after a particular amount of time, the user assumes the packet has been lost as a result of a collision

[1] Switch-based Ethernet does not need a MAC protocol because issues of congestion become problems at the network and transport layers because the medium is not shared anymore.

and doubles CW (exponential backoff). It then picks a new backoff interval I' uniformly between 1 and the new CW. It now waits for I' idle slots, before attempting to transmit the packet. Again, if there is a collision when the packet is transmitted, it doubles the CW, and this process repeats itself. After the number of collisions for a given packet reaches some upper limit, the MAC layer gives up and reports an error to the end user. If the packet is eventually ACKed after some number of collisions, the process starts afresh for the next packet.

This overly conservative behavior in WiFi (backing off even before transmitting the first time) reflects a desire to avoid collisions at all costs. If two users sense the medium is idle and immediately transmit, there is going to be a collision whether it is WiFi or Ethernet. The cost of a collision is higher in WiFi than Ethernet. Collisions are expensive in WiFi relative to Ethernet because in WiFi the *entire packet* must be transmitted and then ACKed (or not ACKed) to determine whether there has been a collision. In the case of Ethernet, the ability to simultaneously transmit and receive allows a user to check for collisions after transmitting every bit, which in turn allows the user to abort a collided transmission in the middle of the packet. The consequence of this difference between WiFi and Ethernet is the lower collision detection latency in Ethernet relative to WiFi—and hence the desire to avoid collisions at all costs in WiFi.

4 The hidden terminal problem

Carrier sense relies on the ability of users to hear each other's transmissions. For WiFi, there are cases where this is not true. One example of this is a problem called the *hidden terminal* problem. Let's say we have two laptops A and C at two corners of a straight line and an access points B in the middle of this straight line. We'll assume that A and C are within the communication range of B, but A and C are not within the communication range of each other. We'll define the term communication range a bit more precisely later, but broadly, you can think of it as a limit on the distance between two computers that allows them to communicate reliably on a wireless medium.

Now, if A and C follow CSMA/CA, each will pick a backoff interval from the initial (fairly small) CW, and transmit at this time. If the packets transmitted by A and C are both quite large, both the A to B and the C to B tranmissions will take a fair bit of time, increasing the likelihood that they collide. This collision is the result of A and C being hidden from each other and not realizing that they are transmitting to the same receiver B, which is where the collision occurs.

One mechanism that fixes this problem is called RTS/CTS, where each user that wishes to send to an access point (AP) sends a short Request-To-Send (RTS) message, which the AP acknowledges with a short Clear-To-Send (CTS) response. Now, if both users A and C wish to transmit to AP B, they each send an RTS message after the initial backoff interval. Because the RTS and CTS messages are small, the likelihood of a collision between the RTS messages of A and C destined to B is low. B responds with a CTS to one of the two (typically to the one whose RTS message was received first). Let's say B sends a CTS to A. This CTS is broadcasted over the air and the other user C also hears it. It knows that the CTS is intended for A and it decides to stay silent for the duration of time specified in the CTS. At this point, A is free to tranmsit to B without the likelihood of a collison at B.

References

[1] Kumu Networks brings full-duplex to MWC. `http://www.fiercewireless.com/tech/kumu-networks-brings-full-duplex-to-mwc`.

[2] Dinesh Bharadia, Emily McMilin, and Sachin Katti. Full Duplex Radios. In *SIGCOMM*, 2013.

[3] Shyamnath Gollakota and Dina Katabi. Zigzag Decoding: Combating Hidden Terminals in Wireless Networks. In *SIGCOMM*, 2008.