4.1 THE CHANNEL ALLOCATION PROBLEM

The central theme of this chapter is how to allocate a single broadcast channel among competing users. The channel might be a portion of the wireless spectrum in a geographic region, or a single wire or optical fiber to which multiple nodes are connected. It does not matter. In both cases, the channel connects each user to all other users and any user who makes full use of the channel interferes with other users who also wish to use the channel.

We will first look at the shortcomings of static allocation schemes for bursty traffic. Then, we will lay out the key assumptions used to model the dynamic schemes that we examine in the following sections.

4.1.1 Static Channel Allocation

The traditional way of allocating a single channel, such as a telephone trunk, among multiple competing users is to chop up its capacity by using one of the multiplexing schemes we described in Sec. 2.5, such as FDM (Frequency Division Multiplexing). If there are N users, the bandwidth is divided into N equal-sized portions, with each user being assigned one portion. Since each user has a private frequency band, there is now no interference among users. When there is only a small and constant number of users, each of which has a steady stream or a heavy load of traffic, this division is a simple and efficient allocation mechanism. A wireless example is FM radio stations. Each station gets a portion of the FM band and uses it most of the time to broadcast its signal.

However, when the number of senders is large and varying or the traffic is bursty, FDM presents some problems. If the spectrum is cut up into N regions and fewer than N users are currently interested in communicating, a large piece of valuable spectrum will be wasted. And if more than N users want to communicate, some of them will be denied permission for lack of bandwidth, even if some of the users who have been assigned a frequency band hardly ever transmit or receive anything.

Even assuming that the number of users could somehow be held constant at N, dividing the single available channel into some number of static subchannels is

inherently inefficient. The basic problem is that when some users are quiescent, their bandwidth is simply lost. They are not using it, and no one else is allowed to use it either. A static allocation is a poor fit to most computer systems, in which data traffic is extremely bursty, often with peak traffic to mean traffic ratios of 1000:1. Consequently, most of the channels will be idle most of the time.

The poor performance of static FDM can easily be seen with a simple queueing theory calculation. Let us start by finding the mean time delay, T, to send a frame onto a channel of capacity C bps. We assume that the frames arrive randomly with an average arrival rate of λ frames/sec, and that the frames vary in length with an average length of $1/\mu$ bits. With these parameters, the service rate of the channel is μC frames/sec. A standard queueing theory result is

$$T = \frac{1}{\mu C - \lambda}$$

(For the curious, this result is for an "M/M/1" queue. It requires that the randomness of the times between frame arrivals and the frame lengths follow an exponential distribution, or equivalently be the result of a Poisson process.)

In our example, if C is 100 Mbps, the mean frame length, $1/\mu$, is 10,000 bits, and the frame arrival rate, λ, is 5000 frames/sec, then $T = 200$ μsec. Note that if we ignored the queueing delay and just asked how long it takes to send a 10,000-bit frame on a 100-Mbps network, we would get the (incorrect) answer of 100 μsec. That result only holds when there is no contention for the channel.

Now let us divide the single channel into N independent subchannels, each with capacity C/N bps. The mean input rate on each of the subchannels will now be λ/N. Recomputing T, we get

$$T_N = \frac{1}{\mu(C/N) - (\lambda/N)} = \frac{N}{\mu C - \lambda} = NT \qquad (4\text{-}1)$$

The mean delay for the divided channel is N times worse than if all the frames were somehow magically arranged orderly in a big central queue. This same result says that a bank lobby full of ATM machines is better off having a single queue feeding all the machines than a separate queue in front of each machine.

Precisely the same arguments that apply to FDM also apply to other ways of statically dividing the channel. If we were to use time division multiplexing (TDM) and allocate each user every Nth time slot, if a user does not use the allocated slot, it would just lie fallow. The same would hold if we split up the networks physically. Using our previous example again, if we were to replace the 100-Mbps network with 10 networks of 10 Mbps each and statically allocate each user to one of them, the mean delay would jump from 200 μsec to 2 msec.

Since none of the traditional static channel allocation methods work well at all with bursty traffic, we will now explore dynamic methods.

4.1.2 Assumptions for Dynamic Channel Allocation

Before we get to the first of the many channel allocation methods in this chapter, it is worthwhile to carefully formulate the allocation problem. Underlying all the work done in this area are the following five key assumptions:

1. **Independent Traffic**. The model consists of N independent **stations** (e.g., computers, telephones), each with a program or user that generates frames for transmission. The expected number of frames generated in an interval of length Δt is $\lambda \Delta t$, where λ is a constant (the arrival rate of new frames). Once a frame has been generated, the station is blocked and does nothing until the frame has been successfully transmitted.

2. **Single Channel**. A single channel is available for all communication. All stations can transmit on it and all can receive from it. The stations are assumed to be equally capable, though protocols may assign them different roles (e.g., priorities).

3. **Observable Collisions**. If two frames are transmitted simultaneously, they overlap in time and the resulting signal is garbled. This event is called a **collision**. All stations can detect that a collision has occurred. A collided frame must be transmitted again later. No errors other than those generated by collisions occur.

4. **Continuous or Slotted Time**. Time may be assumed continuous, in which case frame transmission can begin at any instant. Alternatively, time may be slotted or divided into discrete intervals (called slots). Frame transmissions must then begin at the start of a slot. A slot may contain 0, 1, or more frames, corresponding to an idle slot, a successful transmission, or a collision, respectively.

5. **Carrier Sense or No Carrier Sense**. With the carrier sense assumption, stations can tell if the channel is in use before trying to use it. No station will attempt to use the channel while it is sensed as busy. If there is no carrier sense, stations cannot sense the channel before trying to use it. They just go ahead and transmit. Only later can they determine whether the transmission was successful.

Some discussion of these assumptions is in order. The first one says that frame arrivals are independent, both across stations and at a particular station, and that frames are generated unpredictably but at a constant rate. Actually, this assumption is not a particularly good model of network traffic, as it is well known that packets come in bursts over a range of time scales (Paxson and Floyd, 1995; and Leland et al., 1994). Nonetheless, **Poisson models**, as they are frequently called, are useful because they are mathematically tractable. They help us analyze

protocols to understand roughly how performance changes over an operating range and how it compares with other designs.

The single-channel assumption is the heart of the model. No external ways to communicate exist. Stations cannot raise their hands to request that the teacher call on them, so we will have to come up with better solutions.

The remaining three assumptions depend on the engineering of the system, and we will say which assumptions hold when we examine a particular protocol.

The collision assumption is basic. Stations need some way to detect collisions if they are to retransmit frames rather than let them be lost. For wired channels, node hardware can be designed to detect collisions when they occur. The stations can then terminate their transmissions prematurely to avoid wasting capacity. This detection is much harder for wireless channels, so collisions are usually inferred after the fact by the lack of an expected acknowledgement frame. It is also possible for some frames involved in a collision to be successfully received, depending on the details of the signals and the receiving hardware. However, this situation is not the common case, so we will assume that all frames involved in a collision are lost. We will also see protocols that are designed to prevent collisions from occurring in the first place.

The reason for the two alternative assumptions about time is that slotted time can be used to improve performance. However, it requires the stations to follow a master clock or synchronize their actions with each other to divide time into discrete intervals. Hence, it is not always available. We will discuss and analyze systems with both kinds of time. For a given system, only one of them holds.

Similarly, a network may have carrier sensing or not have it. Wired networks will generally have carrier sense. Wireless networks cannot always use it effectively because not every station may be within radio range of every other station. Similarly, carrier sense will not be available in other settings in which a station cannot communicate directly with other stations, for example a cable modem in which stations must communicate via the cable headend. Note that the word "carrier" in this sense refers to a signal on the channel and has nothing to do with the common carriers (e.g., telephone companies) that date back to the days of the Pony Express.

To avoid any misunderstanding, it is worth noting that no multiaccess protocol guarantees reliable delivery. Even in the absence of collisions, the receiver may have copied some of the frame incorrectly for various reasons. Other parts of the link layer or higher layers provide reliability.

4.2 MULTIPLE ACCESS PROTOCOLS

Many algorithms for allocating a multiple access channel are known. In the following sections, we will study a small sample of the more interesting ones and give some examples of how they are commonly used in practice.

4.2.1 ALOHA

The story of our first MAC starts out in pristine Hawaii in the early 1970s. In this case, "pristine" can be interpreted as "not having a working telephone system." This did not make life more pleasant for researcher Norman Abramson and his colleagues at the University of Hawaii who were trying to connect users on remote islands to the main computer in Honolulu. Stringing their own cables under the Pacific Ocean was not in the cards, so they looked for a different solution.

The one they found used short-range radios, with each user terminal sharing the same upstream frequency to send frames to the central computer. It included a simple and elegant method to solve the channel allocation problem. Their work has been extended by many researchers since then (Schwartz and Abramson, 2009). Although Abramson's work, called the ALOHA system, used ground-based radio broadcasting, the basic idea is applicable to any system in which uncoordinated users are competing for the use of a single shared channel.

We will discuss two versions of ALOHA here: pure and slotted. They differ with respect to whether time is continuous, as in the pure version, or divided into discrete slots into which all frames must fit.

Pure ALOHA

The basic idea of an ALOHA system is simple: let users transmit whenever they have data to be sent. There will be collisions, of course, and the colliding frames will be damaged. Senders need some way to find out if this is the case. In the ALOHA system, after each station has sent its frame to the central computer, this computer rebroadcasts the frame to all of the stations. A sending station can thus listen for the broadcast from the hub to see if its frame has gotten through. In other systems, such as wired LANs, the sender might be able to listen for collisions while transmitting.

If the frame was destroyed, the sender just waits a random amount of time and sends it again. The waiting time must be random or the same frames will collide over and over, in lockstep. Systems in which multiple users share a common channel in a way that can lead to conflicts are known as **contention** systems.

A sketch of frame generation in an ALOHA system is given in Fig. 4-1. We have made the frames all the same length because the throughput of ALOHA systems is maximized by having a uniform frame size rather than by allowing variable-length frames.

Whenever two frames try to occupy the channel at the same time, there will be a collision (as seen in Fig. 4-1) and both will be garbled. If the first bit of a new frame overlaps with just the last bit of a frame that has almost finished, both frames will be totally destroyed (i.e., have incorrect checksums) and both will have to be retransmitted later. The checksum does not (and should not) distinguish between a total loss and a near miss. Bad is bad.

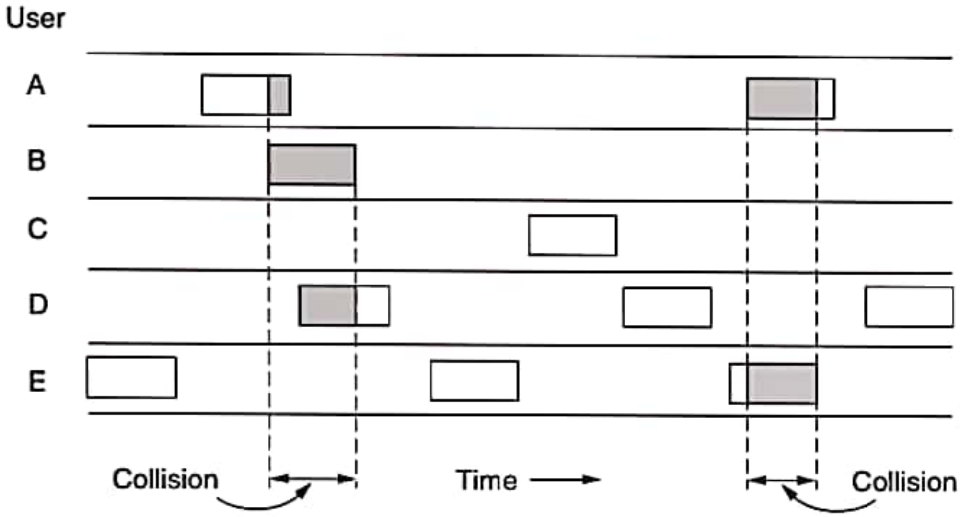

Figure 4-1. In pure ALOHA, frames are transmitted at completely arbitrary times.

An interesting question is: what is the efficiency of an ALOHA channel? In other words, what fraction of all transmitted frames escape collisions under these chaotic circumstances? Let us first consider an infinite collection of users typing at their terminals (stations). A user is always in one of two states: typing or waiting. Initially, all users are in the typing state. When a line is finished, the user stops typing, waiting for a response. The station then transmits a frame containing the line over the shared channel to the central computer and checks the channel to see if it was successful. If so, the user sees the reply and goes back to typing. If not, the user continues to wait while the station retransmits the frame over and over until it has been successfully sent.

Let the "frame time" denote the amount of time needed to transmit the standard, fixed-length frame (i.e., the frame length divided by the bit rate). At this point, we assume that the new frames generated by the stations are well modeled by a Poisson distribution with a mean of N frames per frame time. (The infinite-population assumption is needed to ensure that N does not decrease as users become blocked.) If $N > 1$, the user community is generating frames at a higher rate than the channel can handle, and nearly every frame will suffer a collision. For reasonable throughput, we would expect $0 < N < 1$.

In addition to the new frames, the stations also generate retransmissions of frames that previously suffered collisions. Let us further assume that the old and new frames combined are well modeled by a Poisson distribution, with mean of G frames per frame time. Clearly, $G \geq N$. At low load (i.e., $N \approx 0$), there will be few collisions, hence few retransmissions, so $G \approx N$. At high load, there will be many collisions, so $G > N$. Under all loads, the throughput, S, is just the offered load, G, times the probability, P_0, of a transmission succeeding—that is, $S = GP_0$, where P_0 is the probability that a frame does not suffer a collision.

A frame will not suffer a collision if no other frames are sent within one frame time of its start, as shown in Fig. 4-2. Under what conditions will the

shaded frame arrive undamaged? Let t be the time required to send one frame. If any other user has generated a frame between time t_0 and $t_0 + t$, the end of that frame will collide with the beginning of the shaded one. In fact, the shaded frame's fate was already sealed even before the first bit was sent, but since in pure ALOHA a station does not listen to the channel before transmitting, it has no way of knowing that another frame was already underway. Similarly, any other frame started between $t_0 + t$ and $t_0 + 2t$ will bump into the end of the shaded frame.

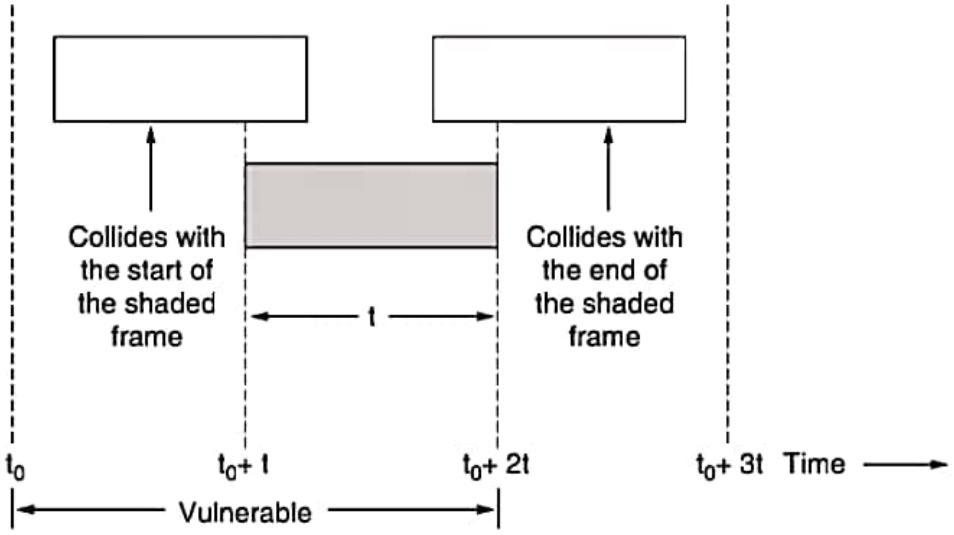

Figure 4-2. Vulnerable period for the shaded frame.

The probability that k frames are generated during a given frame time, in which G frames are expected, is given by the Poisson distribution

$$\Pr[k] = \frac{G^k e^{-G}}{k!} \tag{4-2}$$

so the probability of zero frames is just e^{-G}. In an interval two frame times long, the mean number of frames generated is $2G$. The probability of no frames being initiated during the entire vulnerable period is thus given by $P_0 = e^{-2G}$. Using $S = GP_0$, we get

$$S = Ge^{-2G}$$

The relation between the offered traffic and the throughput is shown in Fig. 4-3. The maximum throughput occurs at $G = 0.5$, with $S = 1/2e$, which is about 0.184. In other words, the best we can hope for is a channel utilization of 18%. This result is not very encouraging, but with everyone transmitting at will, we could hardly have expected a 100% success rate.

Slotted ALOHA

Soon after ALOHA came onto the scene, Roberts (1972) published a method for doubling the capacity of an ALOHA system. His proposal was to divide time into discrete intervals called **slots**, each interval corresponding to one frame. This

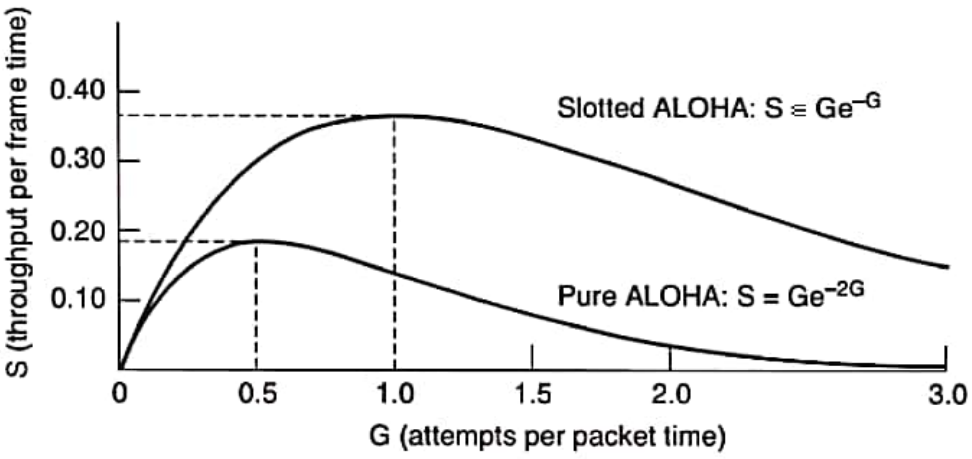

Figure 4-3. Throughput versus offered traffic for ALOHA systems.

approach requires the users to agree on slot boundaries. One way to achieve synchronization would be to have one special station emit a pip at the start of each interval, like a clock.

In Roberts' method, which has come to be known as **slotted ALOHA**—in contrast to Abramson's **pure ALOHA**—a station is not permitted to send whenever the user types a line. Instead, it is required to wait for the beginning of the next slot. Thus, the continuous time ALOHA is turned into a discrete time one. This halves the vulnerable period. To see this, look at Fig. 4-3 and imagine the collisions that are now possible. The probability of no other traffic during the same slot as our test frame is then e^{-G}, which leads to

$$S = Ge^{-G} \tag{4-3}$$

As you can see from Fig. 4-3, slotted ALOHA peaks at $G = 1$, with a throughput of $S = 1/e$ or about 0.368, twice that of pure ALOHA. If the system is operating at $G = 1$, the probability of an empty slot is 0.368 (from Eq. 4-2). The best we can hope for using slotted ALOHA is 37% of the slots empty, 37% successes, and 26% collisions. Operating at higher values of G reduces the number of empties but increases the number of collisions exponentially. To see how this rapid growth of collisions with G comes about, consider the transmission of a test frame. The probability that it will avoid a collision is e^{-G}, which is the probability that all the other stations are silent in that slot. The probability of a collision is then just $1 - e^{-G}$. The probability of a transmission requiring exactly k attempts (i.e., $k - 1$ collisions followed by one success) is

$$P_k = e^{-G}(1 - e^{-G})^{k-1}$$

The expected number of transmissions, E, per line typed at a terminal is then

$$E = \sum_{k=1}^{\infty} kP_k = \sum_{k=1}^{\infty} ke^{-G}(1 - e^{-G})^{k-1} = e^{G}$$

As a result of the exponential dependence of E upon G, small increases in the channel load can drastically reduce its performance.

Slotted ALOHA is notable for a reason that may not be initially obvious. It was devised in the 1970s, used in a few early experimental systems, then almost forgotten. When Internet access over the cable was invented, all of a sudden there was a problem of how to allocate a shared channel among multiple competing users. Slotted ALOHA was pulled out of the garbage can to save the day. Later, having multiple RFID tags talk to the same RFID reader presented another variation on the same problem. Slotted ALOHA, with a dash of other ideas mixed in, again came to the rescue. It has often happened that protocols that are perfectly valid fall into disuse for political reasons (e.g., some big company wants everyone to do things its way) or due to ever-changing technology trends. Then, years later some clever person realizes that a long-discarded protocol solves his current problem. For this reason, in this chapter we will study a number of elegant protocols that are not currently in widespread use but might easily be used in future applications, provided that enough network designers are aware of them. Of course, we will also study many protocols that are in current use as well.