MIT 6.02 **DRAFT** *Lecture Notes*
Last update: April 11, 2011
Comments, questions or bug reports?
Please contact hari at mit.edu

CHAPTER 15
Sharing a Channel: Media Access (MAC) Protocols

There are many communication channels, including radio and acoustic channels, and certain kinds of wired links (coaxial cables), where multiple nodes can all be connected and hear each other's transmissions (either perfectly or with some non-zero probability). This chapter addresses the fundamental question of how such a common communication channel—also called a *shared medium*—can be shared between the different nodes.

There are two fundamental ways of sharing such channels (or media): *time sharing* and *frequency sharing*.[1] The idea in time sharing is to have the nodes coordinate with each other to divide up the access to the medium one at a time, in some fashion. The idea in frequency sharing is to divide up the frequency range available between the different transmitting nodes in a way that there is little or no interference between concurrently transmitting nodes. The methods used here are the same as in frequency division multiplexing, which we described in the previous chapter.

This chapter focuses on time sharing. We will investigate two common ways: *time division multiple access*, or *TDMA*, and *contention protocols*. Both approaches are used in networks today.

These schemes for time and frequency sharing are usually implemented as *communication protocols*. The term *protocol* refers to the rules that govern what each node is allowed to do and how it should operate. Protocols capture the "rules of engagement" that nodes must follow, so that they can collectively obtain good performance. Because these sharing schemes define how multiple nodes should control their access to a shared medium, they are termed *media access (MAC) protocols* or *multiple access protocols*.

Of particular interest to us are contention protocols, so called because the nodes *contend* with each other for the medium without pre-arranging a schedule that determines who should transmit when, or a frequency reservation that guarantees little or no interference. These protocols operate in *laissez faire* fashion: nodes get to send according to their own

[1]There are other ways too, involving codes that allow multiple concurrent transmissions in the same frequency band, with mechanisms to decode the individual communications. We won't study these more advanced ideas here. These ideas are sometimes used in practice, but all real-world systems use a combination of time and frequency sharing.

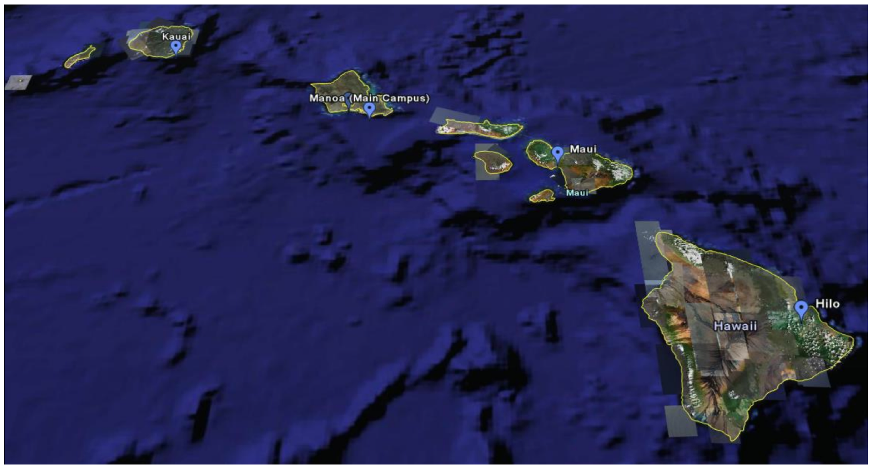

Figure 15-1: The locations of some of the Alohanet's original ground stations are shown in light blue markers.

volition without any external agent telling them what to do. These contention protocols are well-suited for *data* networks, which are characterized by nodes transmitting data in bursts and at variable rates (we will describe the properties of data networks in more detail in a later chapter on packet switching).

In this chapter and the subsequent ones, we will assume that any message is broken up into a set of one or more packets, and a node attempts to send each packet separately over the shared medium.

■ 15.1 Examples of Shared Media

Satellite communications. Perhaps the first example of a shared-medium network deployed for data communication was a satellite network: the *Alohanet* in Hawaii. The Alohanet was designed by a team led by Norm Abramson in the 1960s at the University of Hawaii as a way to connect computers in the different islands together (Figure 15-1). A computer on the satellite functioned as a *switch* to provide connectivity between the nodes on the islands; any packet between the islands had to be first sent over the *uplink* to the switch,[2] and from there over the *downlink* to the desired destination. Both directions used radio communication and the medium was shared. Eventually, this satellite network was connected to the ARPANET (the precursor to today's Internet).

Such satellite networks continue to be used today in various parts of the world, and they are perhaps the most common (though expensive) way to obtain connectivity in the high seas and other remote regions of the world. Figure 15-2 shows the schematic of such a network connecting islands in the Pacific Ocean and used for teleconferencing.

In these satellite networks, the downlink usually runs over a different frequency band from the uplinks, which all share the same frequency band. The different uplinks, however, need to be shared by different concurrent communications from the ground stations to the satellite.

[2]We will study switches in more detail in later lectures.

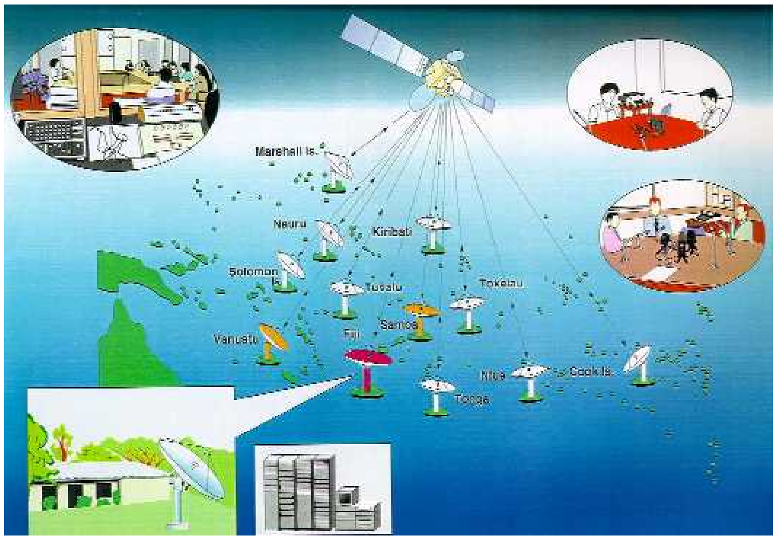

Figure 15-2: A satellite network. The "uplinks" from the ground stations to the satellite form a shared medium. (Picture from `http://vidconf.net/`)

Wireless networks. The most common example of a shared communication medium today, and one that is only increasing in popularity, uses radio. Examples include cellular wireless networks (including standards like EDGE, 3G, and 4G), wireless LANs (such as 802.11, the WiFi standard), and various other forms of radio-based communication. Another example of a communication medium with similar properties is the acoustic channel explored in the 6.02 labs. Broadcast is an inherent property of radio and acoustic communication, especially with so-called omni-directional antennas, which radiate energy in all (or many) different directions. However, radio and acoustic broadcasts are not perfect because of interference and the presence of obstacles on certain paths, so different nodes may correctly receive different parts of any given transmission. This reception is *probabilistic* and the underlying random processes that generate bit errors are hard to model.

Shared bus networks. An example of a wired shared medium is Ethernet, which when it was first developed (and for many years after) used a shared cable to which multiple nodes could be connected. Any packet sent over the Ethernet could be heard by all stations connected physically to the network, forming a perfect shared broadcast medium. If two or more nodes send packets that overlap in time, both packets ended up being garbled and received in error.

Over-the-air radio and television. Even before data communication, many countries in the world had (and still have) radio and television broadcast stations. Here, a relatively small number of transmitters share a frequency range to deliver radio or television content. Because each station was assumed to be active most of the time, the natural approach to sharing is to divide up the frequency range into smaller sub-ranges and allocate each sub-range to a station (frequency division multiplexing).

Given the practical significance of these examples, and the sea change in network access brought about by wireless technologies, developing methods to share a common medium is an important problem.

■ 15.2 Model and Goals

Before diving into the protocols, let's first develop a simple abstraction for the shared medium and more rigorously model the problem we're trying to solve. This abstraction is a reasonable first-order approximation of reality.

We are given a set of N nodes sharing a communication medium. We will assume N is fixed, but the protocols we develop will either continue to work when N varies, or can be made to work with some more effort. Depending on the context, the N nodes may or may not be able to hear each other; in some cases, they may not be able to at all, in some cases, they may, with some probability, and in some cases, they will always hear each other. Each node has some source of data that produces packets. Each packet may be destined for some other node in the network. For now, we will assume that every node has packets destined to one given "master" node in the network. Of course, the master must be capable of hearing every other node, and receiving packets from those nodes. We will assume that the master perfectly receives packets from each node as long as there are no "collisions" (we explain what a "collision" is below).

The model we consider has the following rules:

1. Time is divided into slots of equal length, τ.

2. Each node can send a packet only at the beginning of a slot.

3. All packets are of the same size, and equal to an integral multiple of the *slot length*. In practice, packets will of course be of varying lengths, but this assumption simplifies our analysis and does not affect the correctness of any of the protocols we study.

4. Packets arrive for transmission according to some random process; the protocol should work correctly regardless of the process governing packet arrivals. If two or more nodes send a packet in the same time slot, they are said to *collide*, and *none* of the packets are received successfully. Note that even if only part of a packet encounters a collision, the entire packet is assumed to be lost. This "perfect collision" assumption is an accurate model for wired shared media like Ethernet, but is only a crude approximation of wireless (radio) communication. The reason is that it might be possible for multiple nodes to concurrently transmit data over radio, and depending on the positions of the receivers and the techniques used to decode packets, for the concurrent transmissions to be received successfully.

5. The sending node can discover that a packet transmission collided and may choose to retransmit such a packet.

6. Each node has a queue; any packets waiting to be sent are in the queue. A node with a non-empty queue is said to be *backlogged*.

Performance goals. An important goal is to provide high **throughput**, i.e., to deliver packets successfully at as high a rate as possible, as measured in bits per second. A measure of throughput that is independent of the rate of the channel is the **utilization**, which is defined as follows:

Definition. The **utilization** that a protocol achieves is defined as the **ratio of the total throughput to the maximum data rate of the channel**.

For example, if there are 4 nodes sharing a channel whose maximum bit rate is 10 Megabits/s,[3] and they get throughputs of 1, 2, 2, and 3 Megabits/s, then the utilization is $(1 + 2 + 2 + 3)/10 = 0.8$. Obviously, the utilization is always between 0 and 1. Note that the utilization may be smaller than 1 either because the nodes have enough offered load and the protocol is inefficient, *or* because there isn't enough offered load. By *offered load*, we mean the load presented to the network by a node, or the aggregate load presented to the network by all the nodes. It is measured in bits per second as well.

But utilization alone isn't sufficient: we need to worry about **fairness** as well. If we weren't concerned about fairness, the problem would be quite easy because we could arrange for a particular backlogged node to always send data. If all nodes have enough load to offer to the network, this approach would get high utilization. But it isn't too useful in practice because it would also starve one or more other nodes.

A number of notions of fairness have been developed in the literature, and it's a topic that continues to generate activity and interest. For our purposes, we will use a simple, standard definition of fairness: we will measure the throughput achieved by each node over some time period, T, and say that an allocation with lower standard deviation is "fairer" than one with higher standard deviation. Of course, we want the notion to work properly when the number of nodes varies, so some normalization is needed. We will use the following simplified *fairness index*:

$$F = \frac{(\sum_{i=1}^{N} x_i)^2}{N \sum x_i^2}, \qquad (15.1)$$

where x_i is the throughput achieved by node i and there are N backlogged nodes in all.

Clearly, $1/N \leq F \leq 1$; $F = 1/N$ implies that a single node gets all the throughput, while $F = 1$ implies perfect fairness. We will consider fairness over both the long-term (many thousands of "time slots") and over the short term (tens of slots). It will turn out that in the schemes we study, some schemes will achieve high utilization but poor fairness, and that as we improve fairness, the overall utilization will drop.

The next section discusses Time Division Multiple Access, or TDMA, a scheme that achieves high fairness, but whose utilization may be low when the offered load is non-uniform between the nodes, and is not easy to implement in a fully distributed way without a central coordinator when nodes join and leave dynamically. However, there are practical situations when TDMA works well, and such protocols are used in some cellular wireless networks. Then, we will discuss a variant of the *Aloha* protocol, the first contention MAC protocol that was invented. Aloha forms the basis for many widely used contention protocols, including the ones used in the IEEE 802.11 (WiFi) standard.

[3]In this course, and in most, if not all, of the networking and communications world, "kilo" = 10^3, "mega" = 10^6 and "giga" = 10^9, when talking about network rates, speeds, or throughput. When referring to storage units, however, one needs to be more careful because "kilo", "mega" and "giga" often (but not always) refer to 2^{10}, 2^{20}, and 2^{30}, respectively.

■ 15.3 Time Division Multiple Access (TDMA)

If one had a centralized resource allocator, such as a base station in a cellular network, and a way to ensure some sort of time synchronization between nodes, then a "time division" is not hard to develop. As the name suggests, the goal is to divide time evenly between the N nodes. One way to achieve this goal is to divide time into slots starting from 0 and incrementing by 1, and for each node to be given a unique identifier (ID) in the range $[0, N-1]$.

A simple TDMA protocol uses the following rule:

> If the current time slot is t, then the node with ID i transmits if, and only if, it is backlogged and $t \text{mod} N = i$.

If the node whose turn it is to transmit in time slot t is not backlogged, then that time slot is "wasted".

This TDMA scheme has some good properties. First, it is fair: each node gets the same number of transmission attempts because the protocol provides access to the medium in round-robin fashion among the nodes. The protocol also incurs no packet collisions (assuming it is correctly implemented!): exactly one node is allowed to transmit in any time slot. And if the number of nodes is static, and there is a central coordinator (e.g., a master nodes), this TDMA protocol is simple to implement.

This TDMA protocol does have some drawbacks. First and foremost, if the nodes send data in bursts, alternating between periods when they are backlogged and when they are not, or if the amount of data sent by each node is different, then TDMA under-utilizes the medium. The degree of under-utilization depends on how skewed the traffic pattern; the more the imbalance, the lower the utilization. An "ideal" TDMA scheme would provide equal access to the medium only among currently backlogged nodes, but even in a system with a central master, knowing which nodes are currently backlogged is somewhat challenging. Second, if each node sends packets that are of different sizes (as is the case in practice, though the model we specified above did not have this wrinkle), making TDMA work correctly is more involved. It can still be made to work, but it takes more effort. An important special case is when each node sends packets of the same size, but the size is bigger than a single time slot. This case is not hard to handle, though it requires a little more thinking, and is left as an exercise for the reader.) Third, making TDMA work in a fully distributed way in a system without a central master, and in cases when the number of nodes changes dynamically, is tricky. It can be done, but the protocol quickly becomes more complex than the simple rule stated above.

Contention protocols like Aloha and CSMA don't suffer from these problems, but unlike TDMA, they encounter packet collisions. In general, burst data and skewed workloads favor contention protocols over TDMA. The intuition in these protocols is that we somehow would like to allocate access to the medium fairly, but only among the *backlogged* nodes. Unfortunately, only each node knows with certainty if it is backlogged or not. Our solution is to use *randomization*, a simple but extremely powerful idea; if each backlogged node transmits data with some probability, perhaps we can arrange for the nodes to pick their transmission probabilities to engineer an outcome that has reasonable utilization (throughput) and fairness!

The rest of this chapter describes such randomized contention protocols, starting with the ancestor of them all, Aloha.

■ 15.4 Aloha

The basic variant of the Aloha protocol that we're going to start with is simple, and as follows:

$$\boxed{\textbf{If a node is backlogged, it sends a packet from its queue with probability } p\textbf{.}}$$

From here, until Section 15.6, we will assume that each packet is exactly one slot in length. Such a system is also called **slotted Aloha**.

We have not specified what p is; we will figure that out later, once we analyze the protocol as a function of p. Suppose there are N backlogged nodes and each node uses the same value of p. We can then calculate the utilization of the shared medium as a function of N and p by simply counting the number of slots in which *exactly one node sends a packet*. By definition, a slot with 0 or greater than 1 transmissions does not correspond to a successfully delivered packet, and therefore does not contribute toward the utilization.

If each node sends with probability p, then the probability that exactly one node sends in any given slot is $Np(1-p)^{N-1}$. The reason is that the probability that a specific node sends in the time slot is p, and for its transmission to be successful, all the other nodes should not send. That combined probability is $p(1-p)^{N-1}$. Now, we can pick the successfully transmitting node in N ways, so the probability of exactly one node sending in a slot is $Np(1-p)^{N-1}$.

This quantity is the utilization achieved by the protocol because it is the fraction of slots that count toward useful throughput. Hence,

$$U_{\text{Slotted Aloha}}(p) = Np(1-p)^{N-1}. \tag{15.2}$$

Figure 15-3 shows Eq.(15.2) for $N = 8$ as a function of p. The maximum value of U occurs when $p = 1/N$, and is equal to $(1-\frac{1}{N})^{N-1}$. As $N \to \infty, U \to 1/e \approx 37\%$.[4] This result is an important one: the maximum utilization of slotted Aloha for a large number of backlogged nodes is roughly $1/e$.

37% might seem like a small value (after all, the majority of the slots are being wasted), but notice that the protocol is *extremely simple* and has the virtue that it is hard to botch its implementation! It is fully distributed and requires no coordination or other specific communication between the nodes. That simplicity in system design is worth a lot—oftentimes, it's a very good idea to trade simplicity off for high performance, and worry about optimization only when a specific part of the system is likely to become (or already has become) a bottleneck.

That said, the protocol as described thus far requires a way to set p. Ideally, if each node knew the value of N, setting $p = 1/N$ achieves the maximum. Unfortunately, this isn't as simple as it sounds because N here is the number of *backlogged* nodes that currently have data in their queues. The question then is: how can the nodes pick the best p? We

[4]Here, we use the fact that $\lim_{N\to\infty}(1-1/N)^N = 1/e$. To see why this limit holds, expand the log of the left hand side using a Taylor series: $\log(1-x) = -x - \frac{x^2}{2} - \frac{x^3}{3} - \ldots.$ for $|x| < 1$.

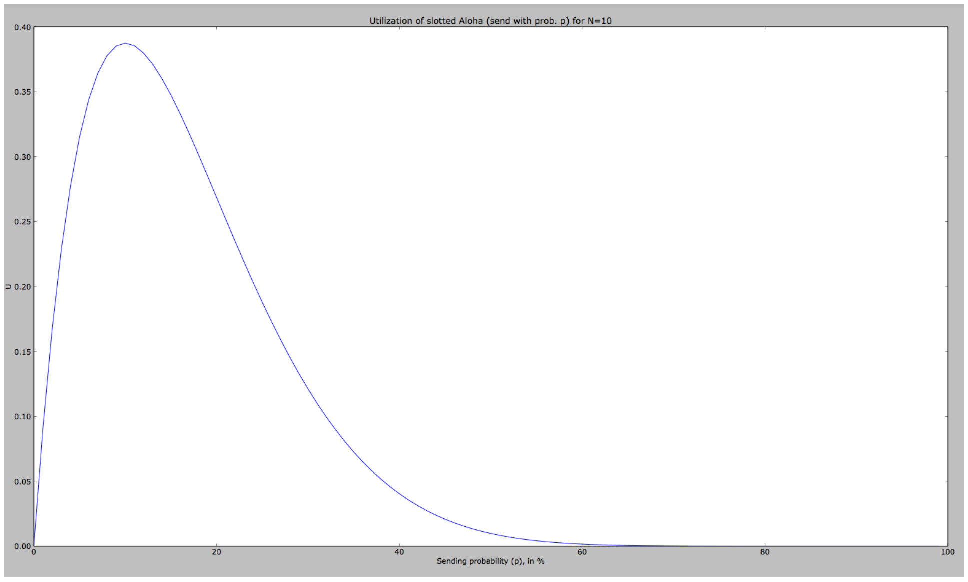

Figure 15-3: The utilization of slotted Aloha as a function of p for $N = 10$. The maximum occurs at $p = 1/N$ and the maximum utilization is $U = (1 - \frac{1}{N})^{N-1}$. As $N \to \infty$, $U \to \frac{1}{e} \approx 37\%$. N doesn't have to be particularly large for the $1/e$ approximation to be close—for instance, when $N = 10$, the maximum utilization is 0.387.

turn to this important question next, because without such a mechanism, the protocol is impractical.

15.5 Stabilizing Aloha: Binary Exponential Backoff

We use a special term for the process of picking a good "p" in Aloha: **stabilization**. In general, in distributed protocols and algorithms, "stabilization" refers to the process by which the method operates around or at a desired operating point. In our case, the desired operating point is around $p = 1/N$, where N is the number of backlogged nodes.

Stabilizing a protocol like Aloha is a difficult problem because the nodes may not be able to directly communicate with each other (or even if they could, the overhead involved in doing so would be significant). Moreover, each node has bursty demands for the medium, and the set of backlogged nodes could change quite rapidly with time. What we need is a "search procedure" by which each node converges toward the best "p".

Fortunately, this search for the right p can be guided by feedback: whether a given packet transmission has been successful or not is invaluable information. In practice, this feedback may be obtained either using an acknowledgment for each received packet from the receiver (as in most wireless networks) or using the ability to directly detect a collision by listening on one's own transmission (as in wired Ethernet). In either case, the feedback has the same form: "yes" or "no", depending on whether the packet was received successfully or not.

Given this feedback, our stabilization strategy at each node is conceptually simple:

1. Maintain the current estimate of p, p_{est}, initialized to some value. (We will talk about initialization later.)

2. If "no", then consider decreasing p.

3. If "yes", then consider increasing p.

This simple-looking structure is at the core of a wide range of distributed network protocols that seek to operate around some desired or optimum value. The devil, of course, is in the details, in that the way in which the increase and decrease rules work depend on the problem and dynamics at hand.

Let's first talk about the decrease rule for our protocol. The intuition here is that because there was a collision, it's likely that the node's current estimate of the best p is too high (equivalently, its view of the number of backlogged nodes is too small). Since the actual number of nodes could be quite a bit larger, a good strategy that quickly gets to the true value is *multiplicative decrease*: reduce p by a factor of 2. Akin to binary search, this method can reach the true probability within a logarithmic number of steps from the current value; absent any other information, it is also the most efficient way to do so.

Thus, the decrease rule is:

$$p \leftarrow p/2 \tag{15.3}$$

This multiplicative decrease scheme has a special name: *binary exponential backoff*. The reason for this name is that if a packet has been unsuccessful k times, the probability with which it is sent decays proportional to 2^{-k}. The "2" is the "binary" part, the k in the exponent is the "exponential" part, and the "backoff" is what the sender is doing in the face of these failures.

To develop an increase rule upon a successful transmission, observe that two factors must be considered: first, the estimate of the number of other backlogged nodes whose queues might have emptied during the time it took us to send our packet successfully, and second, the potential waste of slots that might occur if the increased value of p is too small. In general, if n backlogged nodes contended with a given node x, and x eventually sent its packet, we can expect that some fraction of the n nodes also got their packets through. Hence, the increase in p should at least be multiplicative. p_{max} is a parameter picked by the protocol designer, and must not exceed 1 (obviously).

Thus, one possible increase rule is:

$$p \leftarrow min(2p, p_{\text{max}}). \tag{15.4}$$

Another possible rule is even simpler:

$$p \leftarrow p_{\text{max}}. \tag{15.5}$$

The second rule above isn't unreasonable; in fact, under burst traffic arrivals, it is quite possible for a much smaller number of other nodes to continue to remain backlogged, and in that case resetting to a fixed maximum probability would be a good idea.

For now, let's assume that $p_{\text{max}} = 1$ and use (15.4) to explore the performance of the protocol; one can obtain similar results with (15.5) as well.

■ 15.5.1 Performance

Let's look at how this protocol works in simulation using WSim, a shared medium simulator that you will use in the lab. Running a randomized simulation with $N = 6$ nodes, each generating traffic in a random fashion in such a way that in most slots many of the nodes are backlogged, we see the following result:

```
  Node 0 attempts 335 success 196 coll 139
  Node 1 attempts 1691 success 1323 coll 367
  Node 2 attempts 1678 success 1294 coll 384
  Node 3 attempts 114 success 55 coll 59
  Node 4 attempts 866 success 603 coll 263
  Node 5 attempts 1670 success 1181 coll 489
Time 10000 attempts 6354 success 4652 util 0.47
Inter-node fairness: 0.69
```

Each line starting with "Node" above says what the total number of transmission `attempts` from the specified node was, how many of them were `success`es, and how many of them were `coll`isions. The line starting with "Time" says what the total number of simulated time slots was, and the total number of packet attempts, successful packets (i.e., those without collisions), and the utilization. The last line lists the fairness.

A fairness of 0.69 with six nodes is actually quite poor (in fact, even a value of 0.8 would be considered poor for $N = 6$). Figure 15-4 shows two rows of dots for each node; the top row corresponds to successful transmissions while the bottom one corresponds to collisions. The bar graph in the bottom panel is each node's throughput. Observe how nodes 3 and 0 get very low throughput compared to the other nodes, a sign of significant *long-term unfairness*. In addition, for each node there are long periods of time when both nodes send no packets, because each collision causes their transmission probability to reduce by two, and pretty soon both nodes are made to starve, unable to extricate themselves from this situation. Such "bad luck" tends to happen often because a node that has backed off heavily is competing against a successful backlogged node whose p is a lot higher; hence, the "rich get richer".

How can we overcome this fairness problem? One approach is to set a lower bound on p, something that's a lot smaller than the reciprocal of the largest number of backlogged nodes we expect in the network. In most networks, one can assume such a quantity; for example, we might set the lower bound to 1/128 or 1/1024.

Setting such a bound greatly reduces the long-term unfairness (Figure 15-5) and the corresponding simulation output is as follows:

```
  Node 0 attempts 1516 success 1214 coll 302
  Node 1 attempts 1237 success 964 coll 273
  Node 2 attempts 1433 success 1218 coll 215
  Node 3 attempts 1496 success 1207 coll 289
  Node 4 attempts 1616 success 1368 coll 248
  Node 5 attempts 1370 success 1115 coll 254
Time 10000 attempts 8668 success 7086 util 0.71
Inter-node fairness: 0.99
```

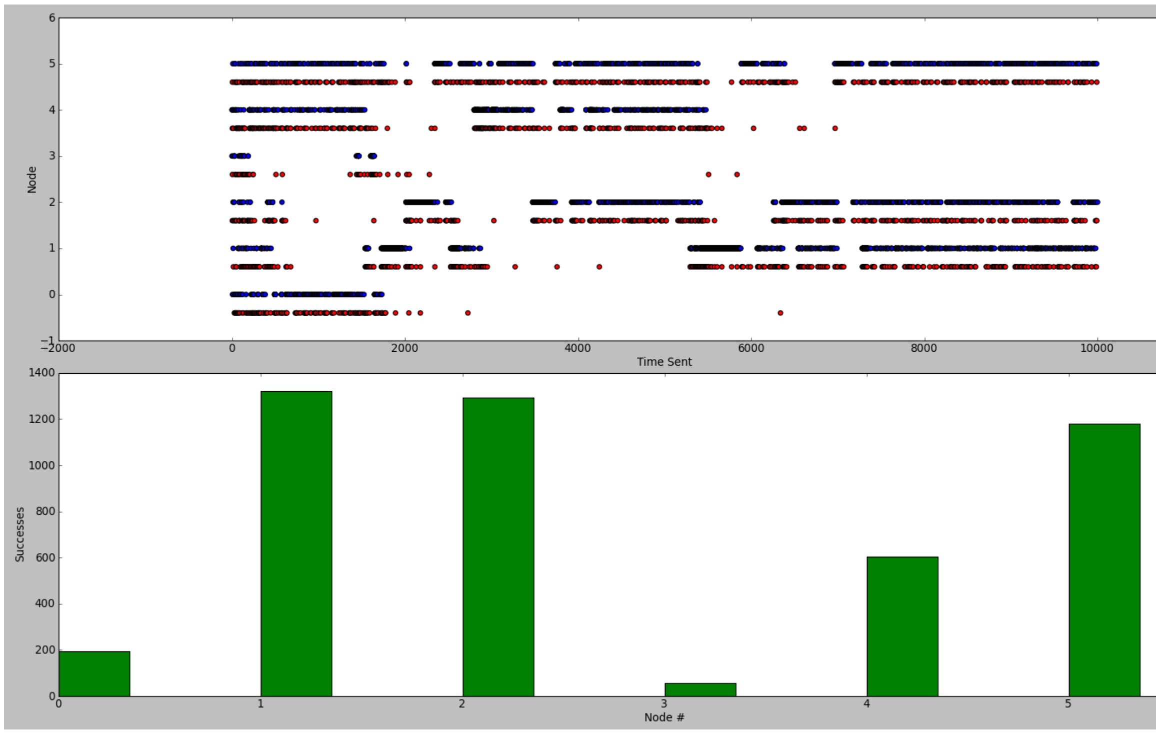

Figure 15-4: For each node, the top row (blue) shows the times at which the node successfully sent a packet, while the bottom row (red) shows collisions. Observe how nodes 3 and 0 are both clobbered getting almost no throughput compared to the other nodes. The reason is that both nodes end up with repeated collisions, and on each collision the probability of transmitting a packet reduces by 2, so pretty soon both nodes are completely shut out. The bottom panel is a bar graph of each node's throughput.

The careful reader will notice something fishy about the simulation output shown above (and also in the output from the simulation where we didn't set a lower bound on p): the reported utilization is 0.71, considerably higher than the "theoretical maximum" of $(1-1/N)^{N-1} = 0.4$ when $N = 6$. What's going on here is more apparent from Figure 15-5, which shows that there are long periods of time where any given node, though backlogged, does not get to transmit. Over time, every node in the experiment encounters times when it is starving, though over time the nodes all get the same share of the medium (fairness is 0.99). If p_{max} is 1 (or close to 1), then a backlogged node that has just succeeded in transmitting its packet will continue to send, while other nodes with smaller values of p end up backing off. This phenomenon is also sometimes called the *capture effect*, manifested by unfairness over time-scales on the order several packets. This behavior is not desirable.

Setting p_{max} to a more reasonable value (less than 1) yields the following:[5]

```
Node 0 attempts 941 success 534 coll 407
Node 1 attempts 1153 success 637 coll 516
Node 2 attempts 1076 success 576 coll 500
Node 3 attempts 1471 success 862 coll 609
Node 4 attempts 1348 success 780 coll 568
```

[5]We have intentionally left the value unspecified because you will investigate how to set it in the lab.

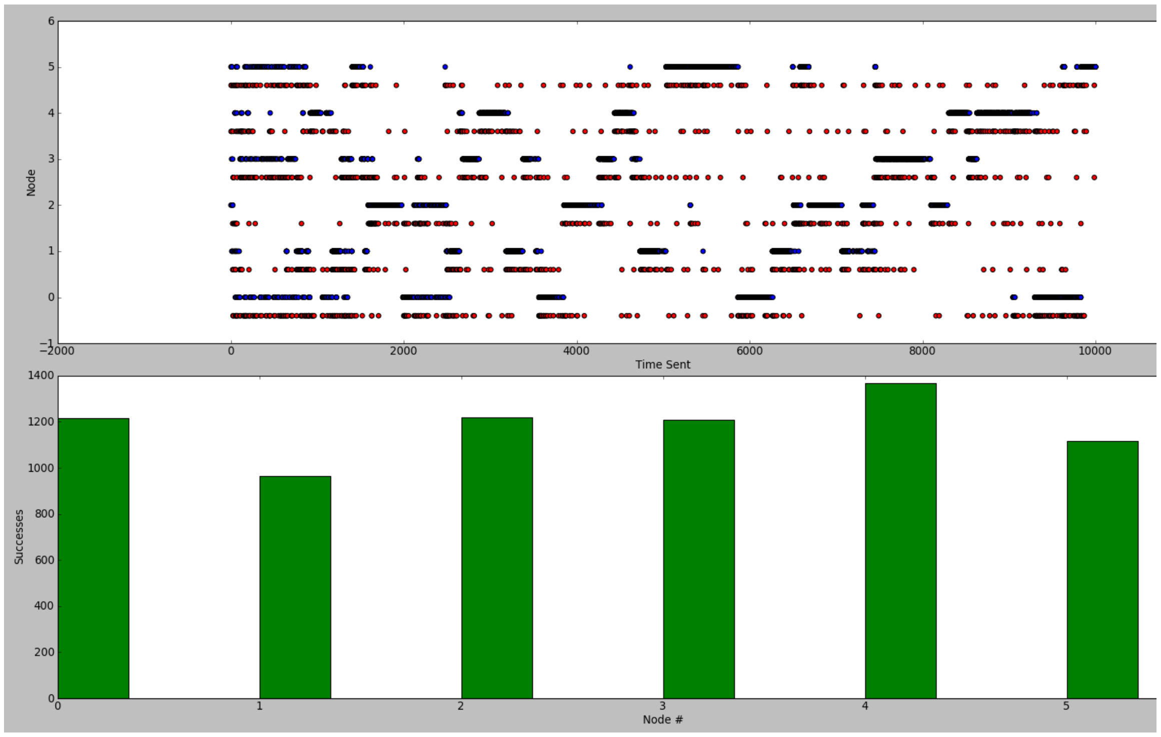

Figure 15-5: Node transmissions and collisions when backlogged v. slot index and each node's throughput (bottom row) when we set a lower bound on each backlogged node's transmission probability. Note the "capture effect" when some nodes hog the medium for extended periods of time, starving others. Over time, however, every node gets the same throughput (fairness is 0.99), but the long periods of inactivity while backlogged is undesirable.

```
  Node 5 attempts 1166 success 683 coll 483
Time 10000 attempts 7155 success 4072 util 0.41
Inter-node fairness: 0.97
```

Figure 15-6 shows the corresponding plot, which has reasonable per-node fairness over both long and short time-scales. The utilization is also close to the value we calculated analytically of $(1 - 1/N)^{N-1}$. Even though the utilization is now lower, the overall result is better because all backlogged nodes get equal share of the medium even over short time scales.

These experiments show the trade-off between achieving both good utilization and ensuring fairness. If our goal were only the former, the problem would be trivial: starve all but one of the backlogged nodes. Achieving a good balance between various notions of fairness and network utilization (throughput) is at the core of many network protocol designs.

■ 15.6 Generalizing to Bigger Packets, and "Unslotted" Aloha

So far, we have looked at perfectly slotted Aloha, which assumes that each packet fits exactly into a slot. But what happens when packets are bigger than a single slot? In fact, one

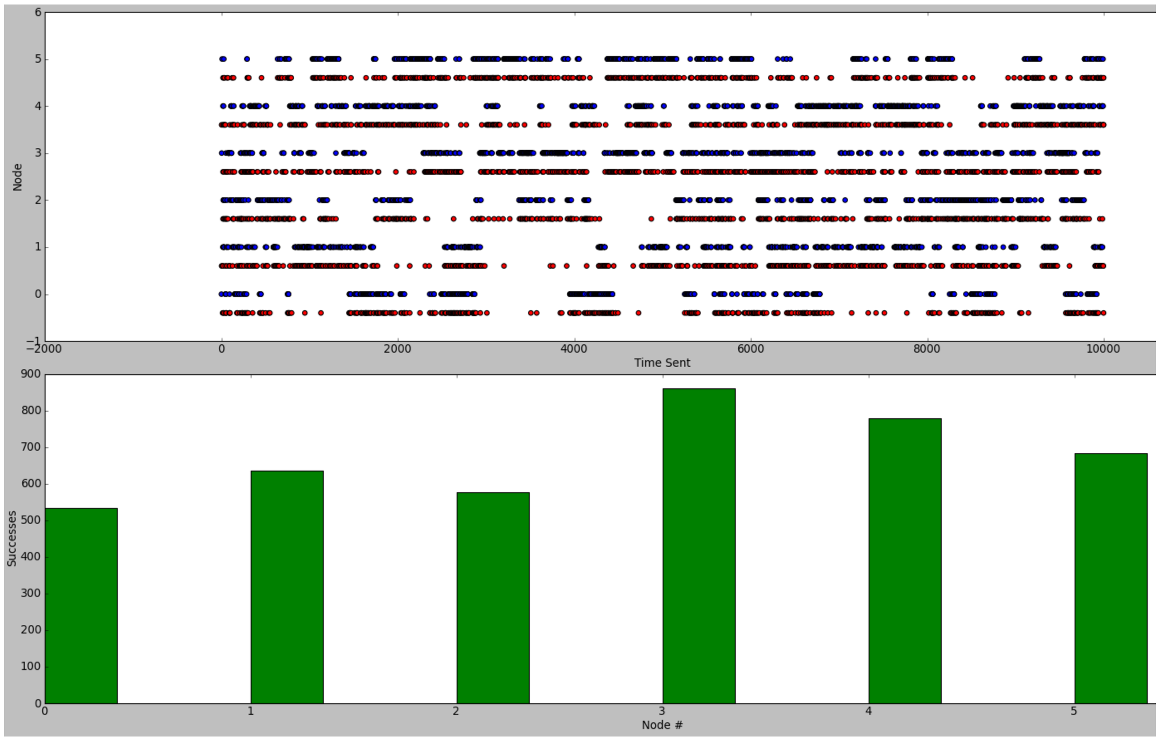

Figure 15-6: Node transmissions and collisions when we set both lower and upper bounds on each backlogged node's transmission probability. Notice that the capture effect is no longer present. The bottom panel is each node's throughput.

might even ask why we need slotting. What happens when nodes just transmit without regard to slot boundaries? In this section, we analyze these issues, starting with packets that span multiple slot lengths. Then, by making a slot length much smaller than a single packet size, we can calculate the utilization of the Aloha protocol where nodes can send without concern for slot boundaries—that variant is also called **unslotted Aloha**.

Note that the pure unslotted Aloha model is one where there are no slots at all, and each node can send a packet any time it wants. However, this model may be approximated by a model where a node sends a packet only at the beginning of a time slot, but each packet is many slots long. When we make the size of a packet large compared to the length of a single slot, we get the unslotted case. We will abuse terminology slightly and use the term *unslotted Aloha* to refer to the case when there are slots, but the packet size is large compared to the slot time.

Suppose each node sends a packet of size T slots. One can then work out the probability of a successful transmission in a network with N backlogged nodes, each attempting to send its packet with probability p whenever it is not already sending a packet. The key insight here is that *any packet whose transmission starts in* $2T-1$ *slots that have any overlap with the current packet can collide.* Figure 15-7 illustrates this point, which we discuss in more detail next.

Suppose that some node sends a packet in some slot. What is the probability that this transmission has no collisions? From Figure 15-7, for this packet to not collide, no other node should start its transmission in $2T-1$ slots. Because p is the probability of a back-

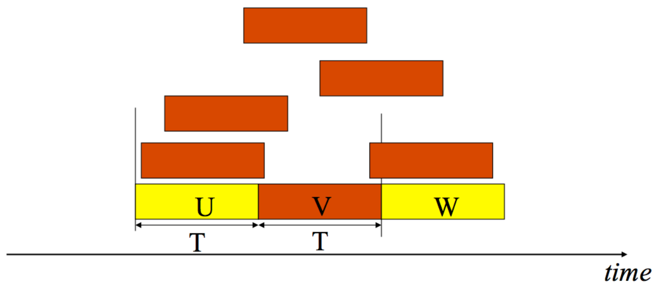

Figure 15-7: Each packet is T slots long. Packet transmissions begin at a slot boundary. In this picture, every packet except U and W collide with V. Given packet V, any other packet sent in any one of $2T-1$ slots—the T slots of V as well as the $T-1$ slots immediately preceding V's transmission—collide with V.

logged node sending a packet in a slot, and there are $N-1$ nodes, this probability is equal to $(1-p)^{(2T-1)(N-1)}$. (There is a bit of an inaccuracy in this expression, which doesn't make a significant material difference to our conclusions below, but which is worth pointing out. This expression assumes that a node sends packet independently in each time slot with probability p. Of course, in practice a node will not be able to send a packet in a time slot if it is sending a packet in the previous time slot, unless the packet being sent in the previous slot has completed. But our assumption in writing this formula is that such "self inteference" is permissible, which can't occur in reality. But it doesn't matter much for our conclusion because we are interested in the utilization when N is large, which means that p would be quite small. Moreover, this formula does represent an accurate *lower bound* on the throughput.)

Now, the transmitting node can be chosen in N ways, and the node has a probability p of sending a packet. Hence, the utilization, U, is equal to

$$\begin{aligned} U &= \text{Throughput}/\text{Maximum rate} \\ &= Np(1-p)^{(2T-1)(N-1)}/(1/T) \\ &= TNp(1-p)^{(2T-1)(N-1)}. \end{aligned} \tag{15.6}$$

For what value of p is U maximized, and what is the maximum value? By differentiating U wrt p and crunching through some algebra, we find that the maximum value, for large N, is $\frac{T}{(2T-1)e}$.

Now, we can look at what happens in the pure unslotted case, when nodes send without regard to slot boundaries. As explained above, the utilization of this scheme is identical to the case when we make the packet size T much larger than 1; i.e., if each packet is large compared to a time slot, then the fact that the model assumes that packets are sent along slot boundaries is irrelevant as far as throughput (utilization) is concerned. The maximum utilization in this case when N is large is therefore equal to $\frac{1}{2e} \approx 0.18$. **Note that this value is one-half of the maximum utilization of pure slotted Aloha where each packet is one slot long.** (We're making this statement for the case when N is large, but it doesn't take N to become all that large for the statement to be roughly true, as we'll see in the lab.)

This result may be surprising at first glance, but it is intuitively quite pleasing. Slotting makes it so two packets destined to collide do so fully. Because partial collisions are just as bad as full ones in our model of the shared medium, forcing a full collision improves utilization. Unslotted Aloha has "twice the window of vulnerability" as slotted Aloha, and in the limit when the number of nodes is large, achieves only one-half the utilization.

■ 15.7 Carrier Sense Multiple Access (CSMA)

So far, we have assumed that no two nodes using the shared medium can hear each other. This assumption is true in some networks, notably the satellite network example mentioned here. Over a wired Ethernet, it is decidedly not true, while over wireless networks, the assumption is sometimes true and sometimes not (if there are three nodes A, B, and C, such that A and C can't usually hear each other, but B can usually hear both A and C, then A and C are said to be *hidden terminals*).

The ability to first *listen* on the medium before attempting a transmission can be used to reduce the number of collisions and improve utilization. The technical term given for this capability is called **carrier sense**: a node, before it attempts a transmission, can listen to the medium to see if the analog voltage or signal level is higher than if the medium were unused, or even attempt to detect if a packet transmission is in progress by processing ("demodulating", a concept we will see in later lectures) a set of samples. Then, if it determines that another packet transmission is in progress, it considers the medium to be *busy*, and *defers* its own transmission attempt until the node considers the medium to be *idle*. The idea is for a node to send only when it believes the medium to be idle.

One can modify the stabilized version of Aloha described above to use CSMA. One advantage of CSMA is that it no longer requires each packet to be one time slot long to achieve good utilization; packets can be larger than a slot duration, and can also vary in length.

Note, however, that in any practical implementation, it will takes some time for a node to detect that the medium is idle after the previous transmission ends, because it takes time to integrate the signal or sample information received and determine that the medium is indeed idle. This duration is called the *detection time* for the protocol. Moreover, multiple backlogged nodes might discover an "idle" medium at the same time; if they both send data, a collision ensues. For both these reasons, CSMA does not achieve 100% utilization, and needs a backoff scheme, though it usually achives higher utilization than stabilized slotted Aloha over a single shared medium. You will investigate this protocol in the lab.

■ 15.8 A Note on Implementation: Contention Windows

In the protocols described so far, each backlogged node sends a packet with probability p, and the job of the protocol is to adapt p in the best possible way. With CSMA, the idea is to send with this probability but only when the medium is idle. In practice, many contention protocols such as the IEEE 802.3 (Ethernet) and 802.11 (WiFi) standards do something a little different: rather than each node transmitting with a probability in each time slot, they use the concept of a **contention window**.

A contention window scheme works as follows. Each node maintains its own current value of the window, which we call CW. CW can vary between CWmin and CWmax; CWmin may be 1 and CWmax may be a number like 1024. When a node decides to transmit, it does so by picking a random number r uniformly in $[1, \text{CW}]$ and sends in time slot $C + r$, where C is the current time slot. If a collision occurs, the node doubles CW; on a successful transmission, a node halves CW (or, as is often the case in practice, directly resets it to CWmin).

You should note that this scheme is similar to the one we studied and analyzed above. The doubling of CW is analogous to halving the transmission probability, and the halving of CW is analogous to doubling the probability (CW has a lower bound; the transmission probability has an upper bound). But there are two crucial differences:

1. Transmissions with a contention window are done according to a uniform probability distribution and not a geometrically distributed one. In the previous case, the a priori probability that the first transmission occurs t slots from now is geometrically distributed; it is $p(1-p)^{t-1}$, while with a contention window, it is equal to $1/\text{CW}$ for $t \in [1, \text{CW}]$ and 0 otherwise. This means that each node is *guaranteed* to attempt a transmission within CW slots, while that is not the case in the previous scheme, where there is always a chance, though exponentially decreasing, that a node may not transmit within any fixed number of slots.

2. The second difference is more minor: each node can avoid generating a random number in each slot; instead, it can generate a random number once per packet transmission attempt.

In the lab, you will implement the key parts of the contention window protocol and experiment with it in conjunction with CSMA. There is one important subtlety to keep in mind while doing this implementation. The issue has to do with how to count the slots before a node decides to transmit. Suppose a node decides that it will transmit x slots from now as long as the medium is idle after x slots; if x includes the busy slots when another node transmits, then multiple nodes may end up trying to transmit in the same time slot after the ending of a long packet transmission from another node, leading to excessive collisions. So it is important to only count down the idle slots; i.e., x should be the number of *idle* slots before the node attempts to transmit its packet (and of course, a node should try to send a packet in a slot only if it believes the medium to be idle in that slot).

■ 15.9 Summary

This lecture discussed the issues involved in sharing a communication medium amongst multiple nodes. We focused on contention protocols, developing ways to make them provide reasonable utilization and fairness. This is what we learned:

1. Good MAC protocols optimize utilization (throughput) and fairness, but must be able to solve the problem in a distributed way. In most cases, the overhead of a central controller node knowing which nodes have packets to send is too high. These protocols must also provide good utilization and fairness under dynamic load.

2. TDMA provides high throughput when all (or most of) the nodes are backlogged and the offered loads is evenly distributed amongst the nodes. When per-node loads are bursty or when different nodes send different amounts of data, TDMA is a poor choice.

3. Slotted Aloha has surprisingly high utilization for such a simple protocol, if one can pick the transmission probability correctly. The probability that maximizes throughput is $1/N$, where N is the number of backlogged nodes, the resulting utilization tends toward $1/e \approx 37\%$, and the fairness is close to 1 if all nodes present the same load. The utilization does remains high even when the nodes present different loads, in contrast to TDMA.

   It is also worth calculating (and noting) how many slots are left idle and how many slots have more than one node transmitting at the same time in slotted Aloha with $p = 1/N$. When N is large, these numbers are $1/e$ and $1 - 2/e \approx 26\%$, respectively. It is interesting that the number of idle slots is the same as the utilization: if we increase p to reduce the number of idle slots, we don't increase the utilization but actually increase the collision rate.

4. Stabilization is crucial to making Aloha practical. We studied a scheme that adjusts the transmission probability, reducing it multiplicatively when a collision occurs and increasing it (either multiplicatively or to a fixed maximum value) when a successful transmission occurs. The idea is to try to converge to the optimum value.

5. A non-zero lower bound on the transmission probability is important if we want to improve fairness, in particular to prevent some nodes from being starved. An upper bound smaller than 1 improves fairness over shorter time scales by alleviating the capture effect, a situation where one or a small number of nodes capture all the transmission attempts for many time slots in succession.

6. Slotted Aloha has double the utilization of unslotted Aloha when the number of backlogged nodes grows. The intuitive reason is that if two packets are destined to collide, the "window of vulnerability" is larger in the unslotted case by a factor of two.

7. A broadcast network that uses packets that are multiple slots in length (i.e., mimicking the unslotted case) can use carrier sense if the medium is a true broadcast medium (or approximately so). In a true broadcast medium, all nodes can hear each other reliably, so they can sense the carrier before transmitting their own packets. By "listening before transmitting" and setting the transmission probability using stabilization, they can reduce the number of collisions and increase utilization, but it is hard (if not impossible) to eliminate all collisions. Fairness still requires bounds on the transmission probability as before.

8. With a contention window, one can make the transmissions from backlogged nodes occur according to a uniform distribution, instead of the geometric distribution imposed by the "send with probability p" schemes. A uniform distribution in a finite window guarantees that each node will attempt a transmission within some fixed number of slots, which is not true of the geometric distribution.

■ Acknowledgments

Mythili Vutukuru provided several useful comments that improved the explanations presented here. Thanks also to Sari Canelake, Katrina LaCurts, and Lavanya Sharan for suggesting helpful improvements, and to Kerry Xing for bug fixes.

■ Problems and Questions

1. We studied TDMA, (stabilized) Aloha, and CSMA protocols in this chapter. In each statement below, assume that the protocols are implemented correctly. Which of these statements is true (more than might be).

   (a) TDMA may have collisions when the size of a packet exceeds one time slot.

   (b) There exists some offered load for which TDMA has lower throughput than slotted Aloha.

   (c) In stabilized Aloha, two nodes have a certain probability of colliding in a time slot. If they actually collide in that slot, then they will experience a lower probability of colliding with each other when they each retry.

   (d) There is **no** workload for which stabilized Aloha achieves a utilization greater that $(1 - 1/N)^{N-1}$ ($\approx 1/e$ for large N) when run for a long period of time.

   (e) In slotted Aloha with stabilization, each node's transmission probability converges to $1/N$, where N is the number of backlogged nodes.

   (f) In a network in which all nodes can hear each other, CSMA will have no collisions when the packet size is larger than one time slot.

2. In the Aloha stabilization protocols we studied, when a node experiences a collision, it decreases its transmission probability, but sets a lower bound, p_{min}. When it transmits successfully, it increases its transmission probability, but sets an upper bound, p_{max}.

   (a) Why would we set a lower bound on p_{min} that is not too close to 0?

   (b) Why would we set p_{max} to be significantly smaller than 1?

   (c) Let N be the average number of backlogged nodes. What happens if we set $p_{\text{min}} >> 1/N$?

3. Alyssa and Ben are all on a shared medium wireless network running a variant of slotted Aloha (all packets are the same size and each packet fits in one slot). Their computers are configured such that Alyssa is 1.5 times as likely to send a packet as Ben. Assume that both computers are backlogged.

   (a) For Alyssa and Ben, what is their probability of transmission such that the utilization of their network is maximized?

   (b) What is the maximum utilization?

4. You have two computers, A and B, sharing a wireless network in your room. The network runs the slotted Aloha protocol with equal-sized packets. You want B to get twice the throughput over the wireless network as A whenever both nodes are backlogged. You configure A to send packets with probability p. What should you set the transmission probability of B to, in order to achieve your throughput goal?

5. Which of the following statements are *always* true for networks with $N > 1$ nodes using correctly implemented versions of unslotted Aloha, slotted Aloha, Time Division Multiple Access (TDMA) and Carrier Sense Multiple Access (CSMA)? Unless otherwise stated, assume that the slotted and unslotted versions of Aloha are stabilized and use the same stabilization method and parameters. Explain your answer for each statement.

   (a) There exists some offered load pattern for which TDMA has lower throughput than slotted Aloha.

   (b) Suppose nodes I, II and III use a fixed probability of $p = 1/3$ when transmitting on a 3-node slotted Aloha network (i.e., $N = 3$). If all the nodes are backlogged then over time the utilization averages out to $1/e$.

   (c) When the number of nodes, N, is large in a stabilized slotted Aloha network, setting $p_{\text{max}} = p_{\text{min}} = 1/N$ will achieve the same utilization as a TDMA network if all the nodes are backlogged.

   (d) Using contention windows with a CSMA implementation guarantees that a packet will be transmitted successfully within some bounded time.

6. Suppose that there are three nodes, A, B, and C, seeking access to a shared medium using slotted Aloha, each using some fixed probability of transmission, where each packet takes one slot to transmit. Assume that the nodes are always backlogged, and that node A has half the probability of transmission as the other two, i.e., $p_A = p$ and $p_B = p_C = 2p$.

   (a) If $p_A = 0.3$, compute the average utilization of the network.

   (b) What value of p_A maximizes the average utilization of the network and what is the corresponding maximum utilization?

7. Ben Bitdiddle sets up a shared medium wireless network with one access point and N client nodes. Assume that the N client nodes are backlogged, each with packets destined for the access point. The access point is also backlogged, with each of its packets destined for some client. The network uses slotted Aloha with each packet fitting exactly in one slot. Recall that each backlogged node in Aloha sends a packet with some probability p. Two or more distinct nodes (whether client or access point) sending in the same slot causes a collision. Ben sets the transmission probability, p, of each client node to $1/N$ and sets the transmission probability of the access point to a value p_a.

   (a) What is the utilization of the network in terms of N and p_a?

(b) Suppose N is large. What value of p_a ensures that the aggregate throughput of packets received successfully by the N clients is the same as the throughput of the packets received successfully by the access point?

8. Consider the same setup as the previous problem, but *only the client nodes are backlogged—the access point has no packets to send.* Each client node sends with probability p (don't assume it is $1/N$).

   Ben Bitdiddle comes up with a cool improvement to the receiver at the access point. If exactly one node transmits, then the receiver works as usual and is able to correctly decode the packet. If exactly two nodes transmit, he uses a method to *cancel the interference* caused by each packet on the other, and is (quite remarkably) able to decode both packets correctly.

   (a) What is the probability, P_2, of *exactly* two of the N nodes transmitting in a slot? Note that we want the probability of *any two* nodes sending in a given slot.
   (b) What is the utilization of slotted Aloha with Ben's receiver modification? Write your answer in terms of N, p, and P_2, where P_2 is defined in the problem above.

9. Imagine a shared medium wireless network with N nodes. Unlike a perfect broadcast network in which all nodes can reliably hear any other node's transmission attempt, nodes in our network hear each other probabilistically. That is, between any two nodes i and j, i can hear j's transmission attempt with some probability p_{ij}, where $0 \leq p_{ij} \leq 1$. Assume that all packets are of the same size and that the time slot used in the MAC protocol is much smaller than the packet size.

   (a) Show a configuration of nodes where the throughput achieved when the nodes all use carrier sense is higher than if they didn't.
   (b) Show a configuration of nodes where the throughput achieved when slotted Aloha without carrier sense is higher than with carrier sense.

10. Token-passing is a variant of a TDMA MAC protocol. Here, the N nodes sharing the medium are numbered $0, 1, \ldots N-1$. The token starts at node 0. A node can send a packet if, and only if, it has the token. When node i with the token has a packet to send, it sends the packet and then passes the token to node $(i+1)$ mod N. If node i with the token does not have a packet to send, it passes the token to node $(i+1)$ mod N. To pass the token, a node broadcasts a token packet on the medium and all other nodes hear it correctly.

    A data packet occupies the medium for time T_d. A token packet occupies the medium for time T_k. If s of the N nodes in the network have data to send when they get the token, what is the utilization of the medium? Note that the bandwidth used to send tokens is pure overhead; the throughput we want corresponds to the rate at which data packets are sent.

11. Alyssa P. Hacker is designing a MAC protocol for a network used by people who: live on a large island, never sleep, never have guests, and are always on-line. Suppose the island's network has N nodes, and the island dwellers always keep **exactly**

some four of these nodes backlogged. The nodes communicate with each other by beaming their data to a satellite in the sky, which in turn broadcasts the data down. If two or more nodes transmit in the same slot, their transmissions collide (the satellite uplink doesn't interfere with the downlink). The nodes on the ground **cannot hear each other**, and each node's packet transmission probability is non-zero. Alyssa uses a slotted protocol with **all packets equal to one slot in length**.

(a) For the slotted Aloha protocol with a **fixed** per-node transmission probability, what is the maximum utilization of this network? (Note that there are N nodes in all, of which some four are constantly backlogged.)

(b) Suppose the protocol is the slotted Aloha protocol, and the each island dweller greedily doubles his node transmission probability on each packet collision (but not exceeding 1). What do you expect the network utilization to be?

(c) In this network, as mentioned above, four of the N nodes are constantly backlogged, but the set of backlogged nodes is not constant. Suppose Alyssa must decide between slotted Aloha with a transmission probability of $1/5$ or time division multiple access (TDMA) among the N nodes. For what N does the expected utilization of this slotted Aloha protocol exceed that of TDMA?

(d) Alyssa implements a stabilization protocol to adapt the node transmission probabilities on collisions and on successful transmissions. She runs an experiment and finds that the measured utilization is 0.5. Ben Bitdiddle asserts that this utilization is too high and that she must have erred in her *measurements*. Explain whether or not it is possible for Alyssa's implementation of stabilization to be consistent with her measured result.

12. Recall the MAC protocol with contention windows from §15.8. Here, each node maintains a contention window, W, and sends a packet t idle time slots after the current slot, where t is an integer picked uniformly in $[1, W]$. Assume that each packet is 1 slot long.

    Suppose there are two backlogged nodes in the network with contention windows W_1 and W_2, respectively ($W_1 \geq W_2$). Suppose that both nodes pick their random value of t at the same time. What is the probability that the two nodes will collide the next time they each transmit?

13. Eager B. Eaver gets a new computer with *two* radios. There are N other devices on the shared medium network to which he connects, but each of the other devices has only one radio. The MAC protocol is slotted Aloha with a packet size equal to 1 time slot. Each device uses a fixed transmission probability, and only one packet can be sent successfully in any time slot. All devices are backlogged.

    Eager persuades you that because he has paid for two radios, his computer has a moral right to get twice the throughput of any other device in the network. You begrudgingly agree.

    Eager develops two protocols:

Protocol A: Each radio on Eager's computer runs its MAC protocol independently. That is, each radio sends a packet with fixed probability p. Each other device on the network sends a packet with probability p as well.

Protocol B: Eager's computer runs a single MAC protocol across its two radios, sending packets with probability $2p$, and alternating transmissions between the two radios. Each other device on the network sends a packet with probability p.

(a) With which protocol, A or B, will Eager achieve higher throughput?

(b) Which of the two protocols would you allow Eager to use on the network so that his expected throughput is double any other device's?

14. Carl Coder implements a simple slotted Aloha-style MAC for his room's wireless network. His room has only two backlogged nodes, A and B. Carl picks a transmission probability of $2p$ for node A and p for node B. Each packet is one time slot long and all transmissions occur at the beginning of a time slot.

    (a) What is the utilization of Carl's network in terms of p?

    (b) What value of p maximizes the utilization of this network, **and** what is the maximum utilization?

    (c) Instead of maximizing the utilization, suppose Carl chooses p so that the throughput achieved by A is **three times** the throughput achieved by B. What is the utilization of his network now?

15. Carl Coder replaces the "send with fixed probability" MAC of the previous problem with one that uses a contention window at each node. He configures node A to use a fixed contention window of W and node B to use a fixed contention window of $2W$. Before a transmission, each node independently picks a random integer t uniformly between 1 and its contention window value, and transmits a packet t time slots from now. Each packet is one time slot long and all transmissions occur at the beginning of a time slot.

    (a) Which node, A or B, has a higher probability of being the next to transmit a packet successfully? (Use intuition, don't calculate!)

    (b) What is the probability that A and B will collide the next time they each transmit?

    (c) Suppose A and B each pick a contention window value at some point in time. What is the probability that A transmits before B successfully on its next transmission *attempt*? Note that this probability is equal to the probability that the value picked by A value is strictly smaller than the value picked by B. (It may be useful to apply the formula $\sum_{i=1}^{n} i = n(n+1)/2$.)

    (d) Suppose there is no collision at the next packet transmission. Calculate the probability that A will transmit before B? Explain why this answer is different from the answer to the previous part. You should be able to obtain the solve this problem using the previous two parts.

(e) None of the previous parts directly answer the question, "What is the probability that A will be the first node to successfully transmit a packet before B?" Explain why.