MIT 6.02 **DRAFT** *Lecture Notes*
Spring 2009 (Last update: April 13, 2009)
Comments, questions or bug reports?
Please contact Hari Balakrishnan (hari@mit.edu) or 6.02-staff@mit.edu

LECTURE 18
Sharing a Common Channel: Media Access Protocols

In this course so far, we have studied two kinds of links that connect computers together: a *point-to-point link*, typically constructed from a wire, and a *shared medium* such as radio, for which we developed frequency division multiplexing. In a point-to-point link, the sharing of the link can be done by the switch to which the link is connected. In contrast, when using a shared medium to connect end points together, we might not be able to solve the problem by having all the sharing logic implemented by a single switch—indeed, there may be no switch at all, if the end points are all peers of each other. More pertinently, even if there were a switch present, the information about the queues on the different nodes is *distributed*, and the overhead of a centralized switch deciding how the medium must be shared must be considered and minimized.

This lecture discusses some ways of sharing a common communication medium, or channel, amongst multiple nodes. We start by giving some examples of shared media. Then, we outline three ways in which one can share the medium: frequency division multiplexing, time division multiplexing, and contention protocols. Each of these ways is actually a family of protocols, and each must provide a way to handle state distributed across the nodes.

Most of this lecture will be about *contention protocols*, which provide a fully distributed solution to the problem, and are used widely in many systems today.

■ 18.1 Examples of Shared Media

Satellite communications. Perhaps the first example of such a network deployed in practice was a satellite network: *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 18-1). A switch on the satellite provided the connectivity between the islands, so that any packet between the islands had to be first sent over the *uplink* to the switch, and from there over the *downlink* to the desired destination. Both directions used radio communication and the medium was shared. Eventually,

Figure 18-1: The locations of some of the Alohanet's original ground stations are shown in light blue markers.

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 18-2 shows the schematic of such a network connecting islands in the Pacific Ocean and used for teleconferencing.

Typically, the downlink runs over a different frequency band from the uplinks. The different uplinks, however, need to be shared by different concurrent communications from the ground stations to the satellite.

Wireless data 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 "EDGE", "3G", "4G", etc.), wireless LANs (such as "802.11", the so-called WiFi standard), and various other forms of "packet radio" communications. Radios are not wires; broadcast is an inherent property of these networks. However, the broadcast isn't perfect because of interference, so different nodes may only receive different parts of any given transmission, usually probabilistically (the underlying random processes are complicated and hard to model).

Shared bus networks. An example is Ethernet, which when it was first developed (and for many years after) used a shared wire to which multiple end points (and switches) 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.

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 channel is an important problem.

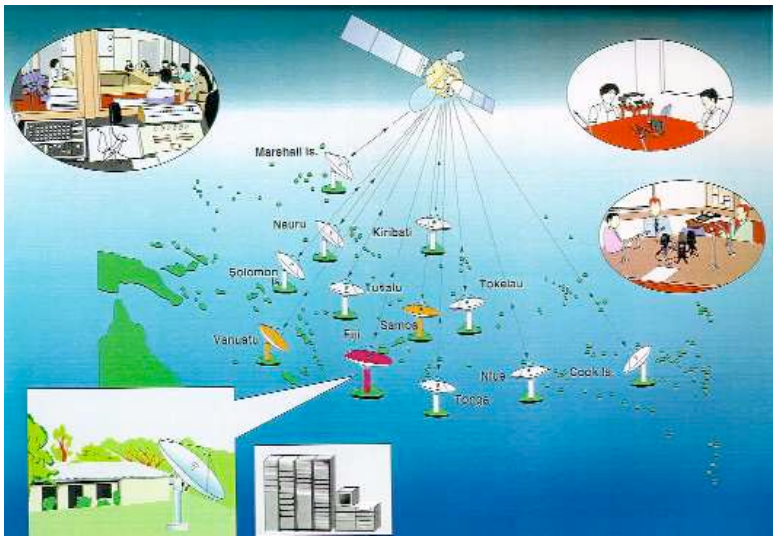

Figure 18-2: A satellite network. The "uplinks" from the ground stations to the satellite form a shared medium. (Picture from `http://vidconf.net/`)

■ 18.2 Approaches to Sharing a Common Channel

Sharing a channel bears some similarity to sharing a link amongst many communications, and similar techniques could be used, such as frequency division multiplexing and time division multiplexing. But there is one crucial difference: the state of the queues is *distributed* amongst the nodes sharing the channel and is not available in a single place as it would when a link is being shared (i.e., there is no switch that knows the precise state of the network). Thus, the best way to view the shared medium is as a distributed queue being processed by a medium at some rate (each successful transmission reduces the queue at some node by one packet). In this model, Little's law continues to apply and we can relate the delay and rate to the average size of the queues across all the nodes.

Both frequency division (whose mechanisms we studied earlier using frequency response and modulation) and time division under-utilize the medium when the different nodes don't all have data to send all the time. When traffic arrives in bursts at the different nodes, and not all at once, a better approach is to avoid wasting bandwidth (frequency bands) or time slots on nodes that don't have any data in their queues. The reasons here are the same as the reasons why packet switching with statistical multiplexing is attractive for sharing links.

On a point-to-point link, a switch simply sends one of the packets from its queue when the link is idle. This method has the property that if only one communication is currently active, that gets all of the link. If more communications become active, the switch is able to share the link between them. We want the same behavior for the shared medium.

Contention protocols are a class of protocols that achieve similar behavior. The term *contention* refers to the fact that nodes *contend* with each other for the medium without pre-arranging a schedule or frequency allocation that determines who goes when. These protocols operate in *laissez faire* fashion: nodes get to send according to their own volition without any external agent telling them what to do.

The term *protocol* refers to the fact that a fully "do what you please" approach is unlikely to work because nodes may simply end up corrupting each others transmissions. Over many shared media, the odds that multiple concurrent transmissions over the medium will succeed is small (we will assume it is zero for this lecture). Hence, we need "rules of engagement" that nodes must follow, so that they can collectively obtain good performance. Such rules of engagement form the protocol; what we will discuss below is one of numerous examples of network protocols.

18.2.1 Goals

Our goal is to develop contention protocols that provide good utilization *and* achieve reasonable fairness between nodes. 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 we also want some sort of "fair" allocation of resources to the different nodes.

There are a large number of notions of fairness that have been developed, and this is a topic that continues to remain of interest. We will use a very simple 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*:

$$F = \frac{(\sum_{i=1}^{N} x_i)^2}{N \sum x_i^2}, \tag{18.1}$$

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

Clearly, F varies between $1/N$ and 1; $1/N$ implies that a single node gets all the throughput, while a value of 1 implies perfect fairness. We will consider fairness over both the long-term (thousands of 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.

We start with a variant of the *Aloha* protocol, the first contention MAC protocol invented.

18.3 Aloha

Let's first develop a simple abstraction for the shared medium. This abstraction is a reasonable, though crude, approximation of reality.

1. All packets are of the same size, and equal to a *slot length*.
2. Time is divided into slots of equal length, τ.
3. Each node can send a packet only at the beginning of a slot.
4. Packets arrive for transmission at random. 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.

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*.

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

> If a node is backlogged, it sends a packet from its queue with probability 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. Before we do that, we need to define what the term "utilization" means.

The utilization that a protocol achieves is defined as the ratio of the total throughput and the maximum data rate of the channel. For example, if there are 4 nodes sharing a channel whose maximum bit rate is 10 Mbits/s, and they get throughputs of 1, 2, 2, and 3 Mbits/s, then the utilization is 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.

So what is the utilization of this "send with probability p" variant of the Aloha protocol? We can answer this question 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 total throughput.

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 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,

$$U_{\text{Aloha}}(p) = Np(1-p)^{N-1}. \tag{18.2}$$

Figure 18-3 shows Eq.(18.2) for $N = 8$ as a function of p. The maximum value of U occurs when $p = 1/N$, and is equal to $1/e \approx 37\%$.[1] This is an important result: the maximum utilization of this variant of Aloha, which is called *slotted Aloha*, is $1/e$.

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. 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.

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

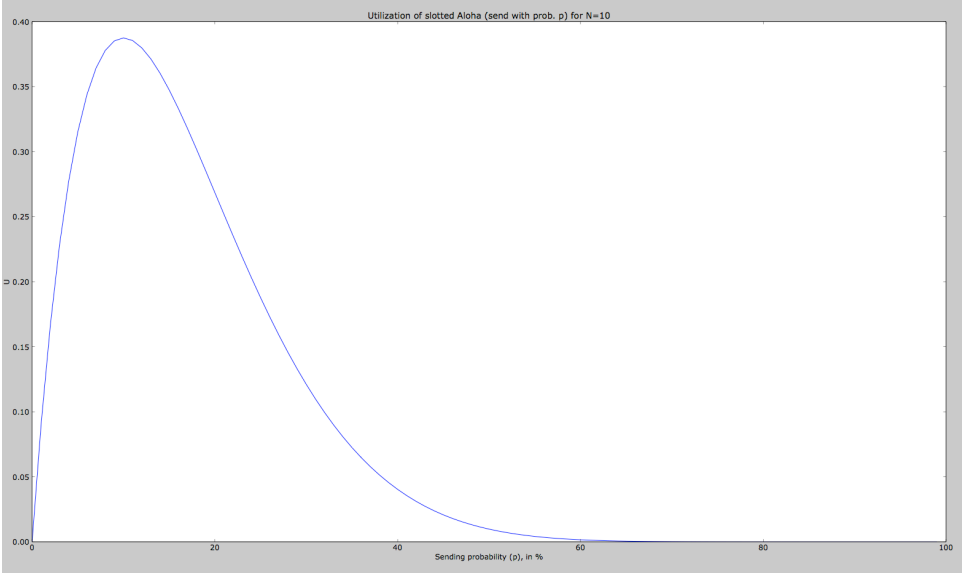

Figure 18-3: The utilization of slotted Aloha as a function of p for $N = 10$. The maximum occurs at $p = 1/N$ and the maximum utilization is $1/e \approx 37\%$.

That said, the protocol as described thus far requires a way to set p. Ideally, if the nodes know N, setting $p = 1/N$ achieves the maximum. The question then is: how can the nodes pick the best p? We turn to this question next.

■ 18.4 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, this term 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 (recall) 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 is quite significant). Moreover, each node has bursty demands for the channel, 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 is 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 decrease p.

3. If "yes", then increase 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:

$$p \leftarrow p/2 \tag{18.3}$$

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 is 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. In fact, under burst traffic arrivals, it is quite possible for a much smaller number of other nodes to continue to remain backlogged. As a result, a reasonable approach would be to reset p to $p_{\max} = 1/\eta$, where η is an estimate of the expected number of *long-term* backlogged nodes. That value might be small, in which case one might set p to 1 (or close to 1), or it might be larger. $p_{\max}$ is usually a parameter picked by the protocol designer. Note that even if this is a bit sub-optimal, not much is lost because the binary exponential backoff procedure mimics binary search to find the current "true" p fairly quickly.

Thus, the increase rule is:

$$p \leftarrow p_{\max}. \tag{18.4}$$

For now, let's assume that $p_{\max} = 1$.

■ 18.4.1 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:

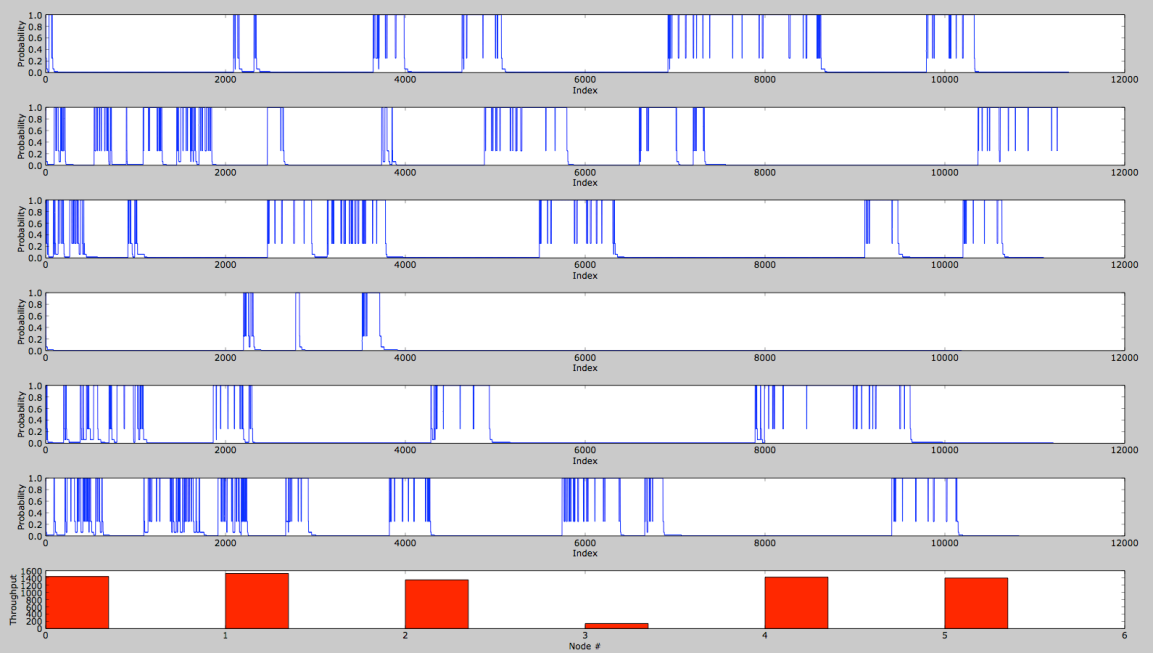

Figure 18-4: Per-node probabilities when backlogged v. slot index and each node's throughput (bottom row). Observe how node 3 is clobbered, a sign of long-term unfairness. In addition, for each node there are long periods of time (x-axis) where p is close to 0 and no packets are sent—a sign of short-term unfairness.

```
  Node 0 attempts 1546 success 1439 coll 107 lat 1932
  Node 1 attempts 1683 success 1524 coll 158 lat 1008
  Node 2 attempts 1471 success 1346 coll 125 lat 1192
  Node 3 attempts 193 success 140 coll 53 lat 2715
  Node 4 attempts 1546 success 1420 coll 126 lat 1338
  Node 5 attempts 1579 success 1396 coll 183 lat 679
Time 10000 attempts 8018 success 7265 util 0.73
```

The chart of the nodes' estimates of p and the throughput of each node is shown in Figure 18-4. Observe how node 3's throughput is clobbered, a sign of long-term unfairness. In addition, for each node there are long periods of time (x-axis) where p is close to 0 and no packets are sent—a sign of short-term unfairness.

The long-term unfairness occurs because node 3 (in this example) ends up with a small probability of transmission from which it is unable to extricate itself. 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".

To solve this problem, we should set a lower bound on p, something that's a lot smaller than 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 this bound greatly reduces the long-term unfairness (Figure 18-5) and the corresponding simulation output is as follows:

```
  Node 0 attempts 1718 success 1473 coll 244 lat 2197
```

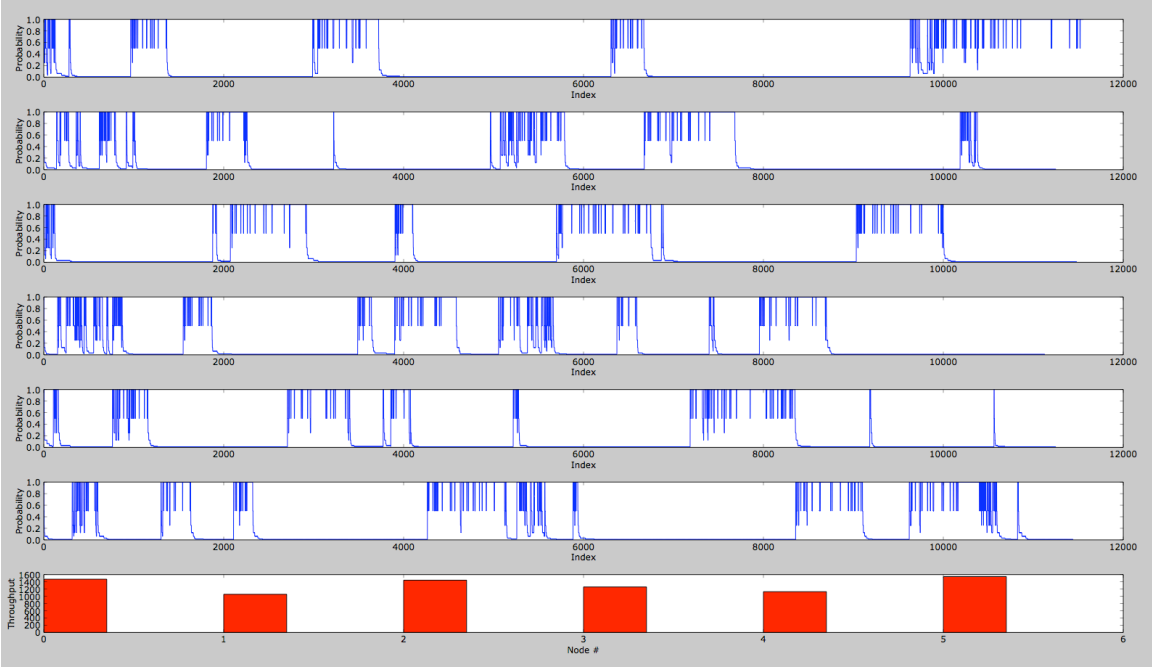

Figure 18-5: Per-node probabilities when backlogged v. slot index and each node's throughput (bottom row) when we set a lower bound on each backlogged node's transmission probability.

```
  Node 1 attempts 1341 success 1055 coll 286 lat 1447
  Node 2 attempts 1667 success 1446 coll 221 lat 1044
  Node 3 attempts 1571 success 1258 coll 313 lat 585
  Node 4 attempts 1356 success 1128 coll 228 lat 1050
  Node 5 attempts 1833 success 1545 coll 288 lat 899
Time 10000 attempts 9486 success 7905 util 0.79
```

The careful reader will notice something fishy about the simulation output shown above (as well as the output from the simulation where we didn't set a lower bound on p): the reported utilization is nearly 0.8, considerably higher than the "theoretical maximum" of $1/e = 0.37$. What's going on here is more apparent from Figure 18-5, which shows that there are long periods of time where any given node, though backlogged, does not get to transmit. Hence, at any given time, each node is only competing with a small number of other nodes, and then going silent. This behavior is not desirable because nodes are starved for long time durations.

Setting an upper bound on p yields the following:

```
  Node 0 attempts 1077 success 655 coll 422 lat 3432
  Node 1 attempts 1071 success 668 coll 403 lat 3080
  Node 2 attempts 1086 success 659 coll 427 lat 2800
  Node 3 attempts 924 success 524 coll 400 lat 3473
  Node 4 attempts 990 success 598 coll 392 lat 2508
  Node 5 attempts 920 success 540 coll 379 lat 3751
Time 10000 attempts 6068 success 3644 util 0.36
```

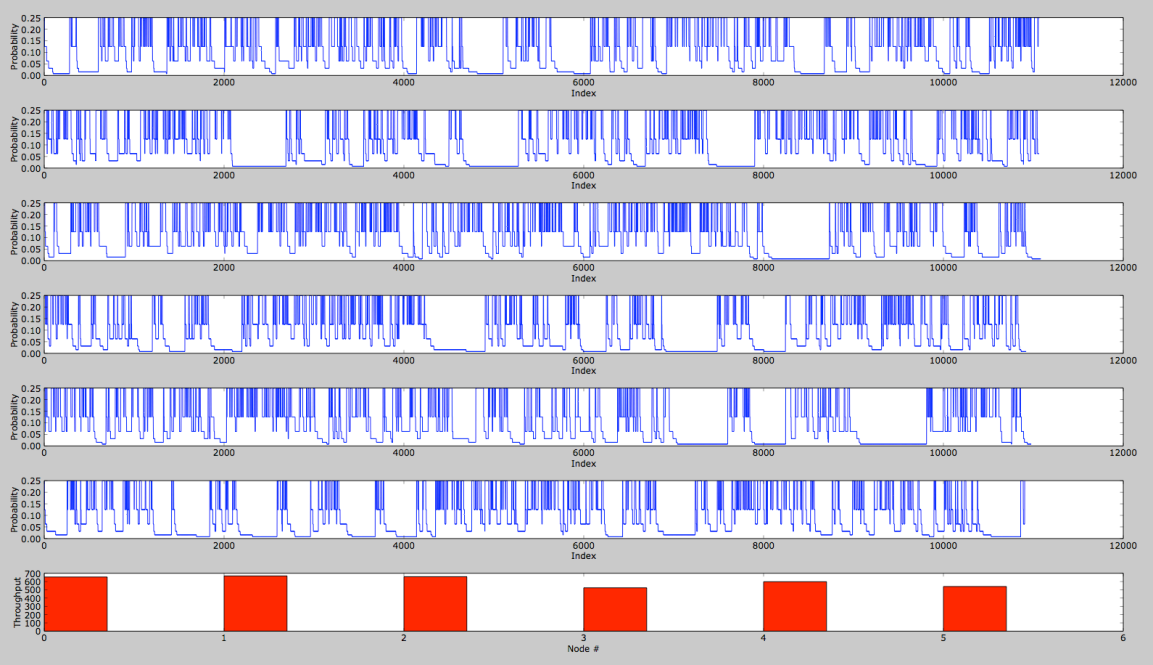

Figure 18-6: Per-node probabilities when backlogged v. slot index and each node's throughput (bottom row) when we set both lower and upper bounds on each backlogged node's transmission probability.

Figure 18-6 shows the corresponding plot, which has reasonable per-node fairness over both long and short time-scales. The utilization is also close to $1/e$, as predicted by theory.

These experiments show the trade-off between achieving both good utilization and ensuring fairness. If our goal were only the latter, 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.

■ 18.5 Slotted v. Unslotted Aloha

So far, we have looked at slotted Aloha, which assumes that each packet fits exactly into a slot. In practice, however, it might not be convenient to break data up into such equal-sized chunks. One might therefore ask whether one needs slotting at all. This section discusses this issue.

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 18-7 illustrates.

Suppose that some node sends a packet in some slot. What is the probability that this transmission has no collisions? From Figure 18-7, for this packet to not collide, no other node should start its transmission in $2T - 1$ slots. Because p is the probability of a node sending a packet in a slot, and there are $N - 1$ nodes, this probability is equal to $(1 -$

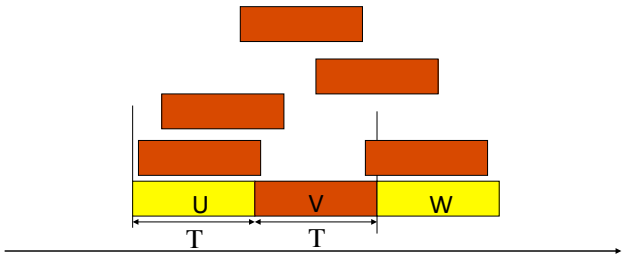

Figure 18-7: Each packet is T slots long. Packet transmissions begin at a slot boundary. In this picture, every packet except U and W collide with V. Given packet V, any other packet sent in any one of $2T-1$ slots—the T slots of V as well as the $T-1$ slots immediately preceding V's transmission—collide with V.

$p)^{(2T-1)(N-1)}$.

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

$$\begin{aligned} U &= \text{Throughput/Maximum rate} \\ &= Np(1-p)^{(2T-1)(N-1)}/(1/T) \\ &= TNp(1-p)^{(2T-1)(N-1)} \end{aligned} \tag{18.5}$$

When is U maximized? By differentiating U wrt p and crunching through some algebra, we find that the maximum value, for large N, is $\frac{T}{(2T-1)e}$. If T is large, i.e., if our packets are many slots long, we get the "unslotted" version of Aloha, whose maximum utilization is $1/2e$. This value is 50% that of slotted Aloha.

This result is intuitively 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.

■ 18.6 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 (in which case the two nodes are said to be *hidden terminals*).

The ability to first *listen* on the channel 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 channel 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 demodulating a set of samples. Then, if the medium is considered busy, the node can *defer* its transmission attempt until the node considers 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 slotting to achieve good utilization; packets can be larger than a slot duration, and can also vary in length.

Note, however, that it takes some time in practice for a node to detect that the medium is idle after the previous transmission ends, because it takes time to intergrate the signal or sample information received. 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 channel.

■ 18.7 Summary

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