1 AN OVERVIEW OF NETWORKS

Somewhere there might be a field of interest in which the order of presentation of topics is well agreed upon.

Computer networking is not it.

There are many interconnections in the field of networking, as in most technical fields, and it is difficult to find an order of presentation that does not involve endless "forward references" to future chapters; this is true even if – as is done here – a largely bottom-up ordering is followed. I have therefore taken here a different approach: this first chapter is a summary of the essentials – LANs, IP and TCP – across the board, and later chapters expand on the material here.

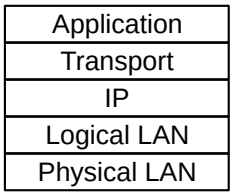
Local Area Networks, or **LANs**, are the "physical" networks that provide the connection between machines within, say, a home, school or corporation. LANs are, as the name says, "local"; it is the **IP**, or Internet Protocol, layer that provides an abstraction for connecting multiple LANs into, well, the Internet. Finally, **TCP** deals with transport and connections and actually sending user data.

This chapter also contains some important other material. The section on **datagram forwarding**, central to packet-based switching and routing, is essential. This chapter also discusses packets generally, congestion, and sliding windows, but those topics are revisited in later chapters. Firewalls and network address translation are also covered here and not elsewhere.

1.1 Layers

These three topics – LANs, IP and TCP – are often called **layers**; they constitute the Link layer, the Internetwork layer, and the Transport layer respectively. Together with the Application layer (the software you use), these form the "**four-layer model**" for networks. A layer, in this context, corresponds strongly to the idea of a programming interface or library, with the understanding that a given layer communicates directly only with the two layers immediately above and below it. An application hands off a chunk of data to the TCP library, which in turn makes calls to the IP library, which in turn calls the LAN layer for actual delivery. An application does *not* interact directly with the IP and LAN layers at all.

The LAN layer is in charge of actual delivery of packets, using LAN-layer-supplied addresses. It is often conceptually subdivided into the "physical layer" dealing with, *eg*, the analog electrical, optical or radio signaling mechanisms involved, and above that an abstracted "logical" LAN layer that describes all the digital – that is, non-analog – operations on packets; see *2.1.4 The LAN Layer*. The physical layer is generally of direct concern only to those designing LAN hardware; the kernel software interface to the LAN corresponds to the logical LAN layer.



This LAN physical/logical division gives us the Internet **five-layer model**. This is less a formal hierarchy as an *ad hoc* classification method. We will return to this below in *1.15 IETF and OSI*, where we will also introduce two more rather obscure layers that complete the **seven**-layer model.

1.2 Data Rate, Throughput and Bandwidth

Any one network connection – *eg* at the LAN layer – has a **data rate**: the rate at which bits are transmitted. In some LANs (*eg* Wi-Fi) the data rate can vary with time. **Throughput** refers to the overall effective transmission rate, taking into account things like transmission overhead, protocol inefficiencies and perhaps even competing traffic. It is generally measured at a higher network layer than the data rate.

The term **bandwidth** can be used to refer to either of these, though we here use it mostly as a synonym for data rate. The term comes from radio transmission, where the width of the frequency band available is proportional, all else being equal, to the data rate that can be achieved.

In discussions about TCP, the term **goodput** is sometimes used to refer to what might also be called "application-layer throughput": the amount of usable data delivered to the receiving application. Specifically, retransmitted data is counted only once when calculating goodput but might be counted twice under some interpretations of "throughput".

Data rates are generally measured in kilobits per second (kbps) or megabits per second (Mbps); the use of the lower-case "b" here denotes bits. In the context of data rates, a kilobit is 10^3 bits (not 2^{10}) and a megabit is 10^6 bits. Somewhat inconsistently, we follow the tradition of using kB and MB to denote data *volumes* of 2^{10} and 2^{20} bytes respectively, with the upper-case B denoting bytes. The newer abbreviations KiB and MiB would be more precise, but the consequences of confusion are modest.

1.3 Packets

Packets are modest-sized buffers of data, transmitted as a unit through some shared set of links. Of necessity, packets need to be prefixed with a **header** containing delivery information. In the common case known as **datagram forwarding**, the header contains a destination **address**; headers in networks using so-called **virtual-circuit** forwarding contain instead an identifier for the *connection*. Almost all networking today (and for the past 50 years) is packet-based, although we will later look briefly at some "circuit-switched" options for voice telephony.

Single and multiple headers

At the LAN layer, packets can be viewed as the imposition of a buffer (and addressing) structure on top of low-level serial lines; additional layers then impose additional structure. Informally, packets are often referred to as **frames** at the LAN layer, and as **segments** at the Transport layer.

The maximum packet size supported by a given LAN (*eg* Ethernet, Token Ring or ATM) is an intrinsic attribute of that LAN. Ethernet allows a maximum of 1500 bytes of data. By comparison, TCP/IP packets originally often held only 512 bytes of data, while early Token Ring packets could contain up to 4 kB of data. While there are proponents of very large packet sizes, larger even than 64 kB, at the other extreme the ATM (Asynchronous Transfer Mode) protocol uses 48 bytes of data per packet, and there are good reasons for believing in modest packet sizes.

One potential issue is how to forward packets from a large-packet LAN to (or through) a small-packet LAN; in later chapters we will look at how the IP (or Internet Protocol) layer addresses this.

Generally each layer adds its own header. Ethernet headers are typically 14 bytes, IP headers 20 bytes, and TCP headers 20 bytes. If a TCP connection sends 512 bytes of data per packet, then the headers amount to 10% of the total, a not-unreasonable overhead. For one common Voice-over-IP option, packets contain 160 bytes of data and 54 bytes of headers, making the header about 25% of the total. Compressing the 160 bytes of audio, however, may bring the data portion down to 20 bytes, meaning that the headers are now 73% of the total; see *20.11.4 RTP and VoIP*.

In datagram-forwarding networks the appropriate header will contain the address of the destination and perhaps other delivery information. Internal nodes of the network called **routers** or **switches** will then try to ensure that the packet is delivered to the requested destination.

The concept of packets and packet switching was first introduced by Paul Baran in 1962 (*[PB62]*). Baran's primary concern was with network survivability in the event of node failure; existing centrally switched protocols were vulnerable to central failure. In 1964, Donald Davies independently developed many of the same concepts; it was Davies who coined the term "packet".

It is perhaps worth noting that packets are buffers built of 8-bit *bytes*, and all hardware today agrees what a byte is (hardware agrees *by convention* on the order in which the bits of a byte are to be transmitted). 8-bit bytes are universal now, but it was not always so. Perhaps the last great non-byte-oriented hardware platform, which did indeed overlap with the Internet era broadly construed, was the DEC-10, which had a 36-bit word size; a word could hold five 7-bit ASCII characters. The early Internet specifications introduced the term **octet** (an 8-bit byte) and required that packets be sequences of octets; non-octet-oriented hosts had to be able to convert. Thus was chaos averted. Note that there are still byte-oriented data issues; as one example, binary integers can be represented as a sequence of bytes in either *big-endian* or *little-endian* byte order (*11.1.5 Binary Data*). **RFC 1700** specifies that Internet protocols use big-endian byte order, therefore sometimes called network byte order.

1.4 Datagram Forwarding

In the datagram-forwarding model of packet delivery, packet headers contain a destination address. It is up to the intervening switches or routers to look at this address and get the packet to the correct destination.

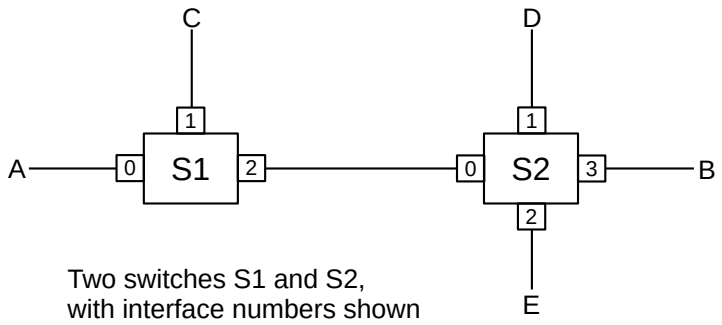
In datagram forwarding this is achieved by providing each switch with a **forwarding table** of ⟨destination,next_hop⟩ pairs. When a packet arrives, the switch looks up the destination address (presumed globally unique) in its forwarding table and finds the **next_hop** information: the immediate-neighbor address to which – or interface by which – the packet should be forwarded in order to bring it one step closer

to its final destination. The next_hop value in a forwarding table is a single entry; each switch is responsible for only one step in the packet's path. However, if all is well, the network of switches will be able to deliver the packet, one hop at a time, to its ultimate destination.

The "destination" entries in the forwarding table do not have to correspond exactly with the packet destination addresses, though in the examples here they do, and they do for Ethernet datagram forwarding. However, for IP routing, the table "destination" entries will correspond to **prefixes** of IP addresses; this leads to a huge savings in space. The fundamental requirement is that the switch can perform a lookup operation, using its forwarding table and the destination address in the arriving packet, to determine the next hop.

Just how the forwarding table is built is a question for later; we will return to this for Ethernet switches in *2.4.1 Ethernet Learning Algorithm* and for IP routers in *9 Routing-Update Algorithms*. For now, the forwarding tables may be thought of as created through initial configuration.

In the diagram below, switch S1 has interfaces 0, 1 and 2, and S2 has interfaces 0,1,2,3. If A is to send a packet to B, S1 must have a forwarding-table entry indicating that destination B is reached via its interface 2, and S2 must have an entry forwarding the packet out on interface 3.



A complete forwarding table for S1, using interface numbers in the `next_hop` column, would be:

S1	
destination	next_hop
A	0
C	1
B	2
D	2
E	2

The table for S2 might be as follows, where we have consolidated destinations A and C for visual simplicity.

S2	
destination	next_hop
A,C	0
D	1
E	2
B	3

In the network diagrammed above, all links are point-to-point, and so each interface corresponds to the unique immediate neighbor reached by that interface. We can thus replace the interface entries in the `next_hop` column with the name of the corresponding **neighbor**. For human readers, using neighbors in the `next_hop` column is usually much more readable. S1's table can now be written as follows (with consolidation of the entries for B, D and E):

S1	
destination	next_hop
A	A
C	C
B,D,E	S2

A central feature of datagram forwarding is that each packet is forwarded "in isolation"; the switches involved do not have any awareness of any higher-layer logical connections established between endpoints. This is also called **stateless** forwarding, in that the forwarding tables have no per-connection state. **RFC 1122** put it this way (in the context of IP-layer datagram forwarding):

> To improve robustness of the communication system, gateways are designed to be stateless, forwarding each IP datagram independently of other datagrams. As a result, redundant paths can be exploited to provide robust service in spite of failures of intervening gateways and networks.

The fundamental alternative to datagram forwarding is **virtual circuits**, *3.4 Virtual Circuits*. In virtual-circuit networks, each router maintains state about each connection passing through it; different connections can be routed differently. If packet forwarding depends, for example, on per-connection information – *eg* both TCP port numbers – it is not datagram forwarding. (That said, it arguably still *is* datagram forwarding if web traffic – to TCP port 80 – is forwarded differently than all other traffic, because that rule does not depend on the specific connection.)

Datagram forwarding is sometimes allowed to use other information beyond the destination address. In theory, IP routing can be done based on the destination address and some **quality-of-service** information, allowing, for example, different routing to the same destination for high-bandwidth bulk traffic and for low-latency real-time traffic. In practice, most Internet Service Providers (ISPs) ignore user-provided quality-of-service information in the IP header, except by prearranged agreement, and route only based on the destination.

By convention, switching devices acting at the LAN layer and forwarding packets based on the LAN address are called **switches** (or, originally, bridges; some still prefer that term), while such devices acting at the IP layer and forwarding on the IP address are called **routers**. Datagram forwarding is used both by Ethernet switches and by IP routers, though the destinations in Ethernet forwarding tables are individual nodes while the destinations in IP routers are entire *networks* (that is, sets of nodes).

In IP routers within end-user sites it is common for a forwarding table to include a catchall **default** entry, matching any IP address that is nonlocal and so needs to be routed out into the Internet at large. Unlike the consolidated entries for B, D and E in the table above for S1, which likely would have to be implemented as actual separate entries, a default entry is a single record representing where to forward the packet if no other destination match is found. Here is a forwarding table for S1, above, with a default entry replacing the last three entries:

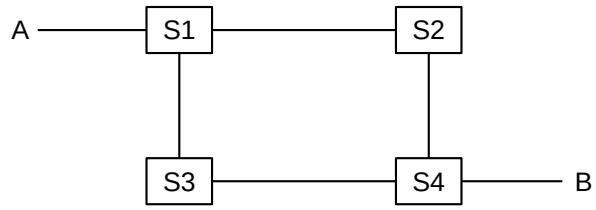
S1	
destination	next_hop
A	0
C	1
default	2

Default entries make sense only when we can tell by looking at an address that it does not represent a nearby node. This is common in IP networks because an IP address encodes the destination network, and routers generally know all the local networks. It is however rare in Ethernets, because there is generally no correlation between Ethernet addresses and locality. If S1 above were an Ethernet switch, and it had some means of knowing that interfaces 0 and 1 connected directly to individual hosts, not switches – and S1 knew the addresses of these hosts – then making interface 2 a default route would make sense. In practice, however, Ethernet switches do not know what kind of device connects to a given interface.

1.5 Topology

In the network diagrammed in the previous section, there are no loops; graph theorists might describe this by saying the network graph is **acyclic**, or is a **tree**. In a loop-free network there is a unique path between any pair of nodes. The forwarding-table algorithm has only to make sure that every destination appears in the forwarding tables; the issue of choosing between alternative paths does not arise.

However, if there are no loops then there is no **redundancy**: any broken link will result in partitioning the network into two pieces that cannot communicate. All else being equal (which it is not, but never mind for now), redundancy is a good thing. However, once we start including redundancy, we have to make decisions among the multiple paths to a destination. Consider, for a moment, the following network:



Should S1 list S2 or S3 as the next_hop to B? Both paths A–S1–**S2**–S4–B and A–S1–**S3**–S4–B get there. There is no right answer. Even if one path is "faster" than the other, taking the slower path is not exactly wrong (especially if the slower path is, say, less expensive). Some sort of protocol must exist to provide a mechanism by which S1 can make the choice (though this mechanism might be as simple as choosing to route via the first path discovered to the given destination). We also want protocols to make sure that, if S1 reaches B via S2 and the S2–S4 link fails, then S1 will switch over to the still-working S1–S3–S4–B route.

As we shall see, many LANs (in particular Ethernet) prefer "tree" networks with no redundancy, while IP has complex protocols in support of redundancy (*9 Routing-Update Algorithms*).

1.5.1 Traffic Engineering

In some cases the decision above between routes A–S1–S2–S4–B and A–S1–S3–S4–B might be of material significance – perhaps the S2–S4 link is slower than the others, or is more congested. We will use the term **traffic engineering** to refer to any intentional selection of one route over another, or any elevation of the priority of one class of traffic. The route selection can either be directly intentional, through configuration, or can be implicit in the selection or tuning of algorithms that then make these route-selection choices automatically. As an example of the latter, the algorithms of *9.1 Distance-Vector Routing-Update Algorithm* build forwarding tables on their own, but those tables are greatly influenced by the administrative assignment of link costs.

With pure datagram forwarding, used at either the LAN or the IP layer, the path taken by a packet is determined solely by its destination, and traffic engineering is limited to the choices made between alternative paths. We have already, however, suggested that datagram forwarding can be extended to take quality-of-service information into account; this may be used to have voice traffic – with its relatively low bandwidth but intolerance for delay – take an entirely different path than bulk file transfers. Alternatively, the network manager may simply assign voice traffic a higher priority, so it does not have to wait in queues behind file-transfer traffic.

The quality-of-service information may be set by the end-user, in which case an ISP may wish to recognize it only for designated users, which in turn means that the ISP will implicitly use the traffic source when making routing decisions. Alternatively, the quality-of-service information may be set by the ISP itself, based on its best guess as to the application; this means that the ISP may be using packet size, port number (*1.12 Transport*) and other contents as part of the routing decision. For some explicit mechanisms supporting this kind of routing, see *9.6 Routing on Other Attributes*.

At the LAN layer, traffic-engineering mechanisms are historically limited, though see *2.8 Software-Defined Networking*. At the IP layer, more strategies are available; see *20 Quality of Service*.

1.6 Routing Loops

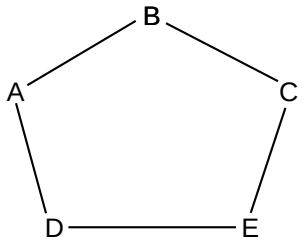
A potential drawback to datagram forwarding is the possibility of a **routing loop**: a set of entries in the forwarding tables that cause some packets to circulate endlessly. For example, in the previous picture we would have a routing loop if, for (nonexistent) destination C, S1 forwarded to S2, S2 forwarded to S4, S4 forwarded to S3, and S3 forwarded to S1. A packet sent to C would not only not be delivered, but in circling endlessly it might easily consume a large majority of the bandwidth. Routing loops typically arise because the creation of the forwarding tables is often "distributed", and there is no global authority to detect inconsistencies. Even when there is such an authority, temporary routing loops can be created due to notification delays.

Routing loops can also occur in networks where the underlying link topology is loop-free; for example, in the previous diagram we could, again for destination C, have S1 forward to S2 and S2 forward back to S1. We will refer to such a case as a **linear** routing loop.

All datagram-forwarding protocols need some way of detecting and avoiding routing loops. Ethernet, for example, avoids nonlinear routing loops by disallowing loops in the underlying network topology, and avoids linear routing loops by not having switches forward a packet back out the interface by which it arrived. IP provides for a one-byte "Time to Live" (TTL) field in the IP header; it is set by the sender and decremented

by 1 at each router; a packet is discarded if its TTL reaches 0. This limits the number of times a wayward packet can be forwarded to the initial TTL value, typically 64.

In datagram routing, a switch is responsible only for the next hop to the ultimate destination; if a switch has a complete path in mind, there is no guarantee that the next_hop switch or any other downstream switch will continue to forward along that path. Misunderstandings can potentially lead to routing loops. Consider this network:



D might feel that the best path to B is D–E–C–B (perhaps because it believes the A–D link is to be avoided). If E similarly decides the best path to B is E–D–A–B, and if D and E both choose their next_hop for B based on these best paths, then a linear routing loop is formed: D routes to B via E and E routes to B via D. Although each of D and E have identified a usable *path*, that path is not in fact followed. Moral: successful datagram routing requires cooperation and a consistent view of the network.

1.7 Congestion

Switches introduce the possibility of congestion: packets arriving faster than they can be sent out. This can happen with just two interfaces, if the inbound interface has a higher bandwidth than the outbound interface; another common source of congestion is traffic arriving on multiple inputs and all destined for the same output.

Whatever the reason, if packets are arriving for a given outbound interface faster than they can be sent, a queue will form for that interface. Once that queue is full, packets will be **dropped**. The most common strategy (though not the only one) is to drop any packets that arrive when the queue is full.

The term "congestion" may refer either to the point where the queue is just beginning to build up, or to the point where the queue is full and packets are lost. In their paper *[CJ89]*, Chiu and Jain refer to the first point as the **knee**; this is where the slope of the load vs throughput graph flattens. They refer to the second point as the **cliff**; this is where packet losses may lead to a precipitous decline in throughput. Other authors use the term **contention** for knee-congestion.

In the Internet, most packet losses are due to congestion. This is not because congestion is especially bad (though it can be, at times), but rather that other types of losses (*eg* due to packet corruption) are insignificant by comparison.

When to Upgrade?

Deciding when a network really *does* have insufficient bandwidth is not a technical issue but an economic one. The number of customers may increase, the cost of bandwidth may decrease or customers may simply be willing to pay more to have data transfers complete in less time; "customers" here can be

external or in-house. Monitoring of links and routers for congestion can, however, help determine exactly what *parts* of the network would most benefit from upgrade.

We emphasize that the presence of congestion does *not* mean that a network has a shortage of bandwidth. Bulk-traffic senders (though not real-time senders) attempt to send as fast as possible, and congestion is simply the network's **feedback** that the maximum transmission rate has been reached. For further discussion, including alternative definitions of longer-term congestion, see *[BCL09]*.

Congestion *is* a sign of a problem in real-time networks, which we will consider in *20 Quality of Service*. In these networks losses due to congestion must generally be kept to an absolute minimum; one way to achieve this is to limit the acceptance of new connections unless sufficient resources are available.

1.8 Packets Again

Perhaps the core justification for packets, Baran's concerns about node failure notwithstanding, is that the same link can carry, at different times, different packets representing traffic to different destinations and from different senders. Thus, packets are the key to supporting **shared transmission lines**; that is, they support the **multiplexing** of multiple communications channels over a single cable. The alternative of a separate physical line between every pair of machines grows prohibitively complex very quickly (though **virtual circuits** between every pair of machines in a datacenter are not uncommon; see *3.4 Virtual Circuits*).

From this shared-medium perspective, an important packet feature is the maximum packet size, as this represents the maximum time a sender can send before other senders get a chance. The alternative of unbounded packet sizes would lead to prolonged network unavailability for everyone else if someone downloaded a large file in a single 1 Gigabit packet. Another drawback to large packets is that, if the packet is corrupted, the entire packet must be retransmitted; see *5.3.1 Error Rates and Packet Size*.

When a router or switch receives a packet, it (generally) reads in the entire packet before looking at the header to decide to what next node to forward it. This is known as **store-and-forward**, and introduces a **forwarding delay** equal to the time needed to read in the entire packet. For individual packets this forwarding delay is hard to avoid (though some switches do implement **cut-through** switching to begin forwarding a packet before it has fully arrived), but if one is sending a long train of packets then by keeping multiple packets *en route* at the same time one can essentially eliminate the significance of the forwarding delay; see *5.3 Packet Size*.

Total packet delay from sender to receiver is the sum of the following:

- **Bandwidth delay**, *ie* sending 1000 Bytes at 20 Bytes/millisecond will take 50 ms. This is a per-link delay.
- **Propagation delay** due to the speed of light. For example, if you start sending a packet right now on a 5000-km cable across the US with a propagation speed of 200 m/μsec (= 200 km/ms, about 2/3 the speed of light in vacuum), the first bit will not arrive at the destination until 25 ms later. The bandwidth delay then determines how much after that the entire packet will take to arrive.
- **Store-and-forward delay**, equal to the sum of the bandwidth delays out of each router along the path
- **Queuing delay**, or waiting in line at busy routers. At bad moments this can exceed 1 sec, though that is rare. Generally it is less than 10 ms and often is less than 1 ms. Queuing delay is the only delay component amenable to reduction through careful engineering.

See *5.1 Packet Delay* for more details.

1.9 LANs and Ethernet

A **local-area network**, or LAN, is a system consisting of

- physical links that are, ultimately, serial lines
- common interfacing hardware connecting the hosts to the links
- protocols to make everything work together

We will explicitly assume that every LAN node is able to communicate with every other LAN node. Sometimes this will require the cooperation of intermediate nodes acting as switches.

Far and away the most common type of (wired) LAN is Ethernet, originally described in a 1976 paper by Metcalfe and Boggs *[MB76]*. Ethernet's popularity is due to low cost more than anything else, though the primary reason Ethernet cost is low is that high demand has led to manufacturing economies of scale.

The original Ethernet had a bandwidth of 10 Mbps (megabits per second; we will use lower-case "b" for bits and upper-case "B" for bytes), though nowadays most Ethernet operates at 100 Mbps and gigabit (1000 Mbps) Ethernet (and faster) is widely used in server rooms. (By comparison, as of this writing (2015) the data transfer rate to a typical faster hard disk is about 1000 Mbps.) Wireless ("Wi-Fi") LANs are gaining popularity, and in many settings have supplanted wired Ethernet to end-users.

Many early Ethernet installations were unswitched; each host simply tapped in to one long primary cable that wound through the building (or floor). In principle, two stations could then transmit at the same time, rendering the data unintelligible; this was called a **collision**. Ethernet has several design features intended to minimize the bandwidth wasted on collisions: stations, before transmitting, check to be sure the line is idle, they monitor the line *while* transmitting to detect collisions during the transmission, and, if a collision is detected, they execute a random backoff strategy to avoid an immediate recollision. See *2.1.5 The Slot Time and Collisions*. While Ethernet collisions definitely reduce throughput, in the larger view they should perhaps be thought of as a part of a remarkably inexpensive shared-access mediation protocol.

In unswitched Ethernets every packet is received by every host and it is up to the network card in each host to determine if the arriving packet is addressed to that host. It is almost always possible to configure the card to forward *all* arriving packets to the attached host; this poses a security threat and "password sniffers" that surreptitiously collected passwords via such eavesdropping used to be common.

Password Sniffing

In the fall of 1994 at Loyola University I remotely changed the root password on several CS-department unix machines at the other end of campus, using telnet. I told no one. Within two hours, someone else logged into one of these machines, using the new password, from a host in Europe. Password sniffing was the likely culprit.

Two months later was the so-called "Christmas Day Attack" (*12.10.1 ISNs and spoofing*). One of the hosts used to launch this attack was Loyola's hacked apollo.it.luc.edu. It is unclear the degree to which password sniffing played a role in that exploit.

Due to both privacy and efficiency concerns, almost all Ethernets today are fully switched; this ensures that each packet is delivered only to the host to which it is addressed. One advantage of switching is that it effectively eliminates most Ethernet collisions; while in principle it replaces them with a **queuing** issue, in practice Ethernet switch queues so seldom fill up that they are almost invisible even to network managers (unlike IP router queues). Switching also prevents host-based eavesdropping, though arguably a better solution to this problem is encryption. Perhaps the more significant tradeoff with switches, historically, was that Once Upon A Time they were expensive and unreliable; tapping directly into a common cable was dirt cheap.

Ethernet addresses are six bytes long. Each Ethernet card (or **network interface**) is assigned a (supposedly) unique address at the time of manufacture; this address is burned into the card's ROM and is called the card's **physical** address or **hardware** address or **MAC** (Media Access Control) address. The first three bytes of the physical address have been assigned to the manufacturer; the subsequent three bytes are a serial number assigned by that manufacturer.

By comparison, IP addresses are assigned administratively by the local site. The basic advantage of having addresses in hardware is that hosts automatically know their own addresses on startup; no manual configuration or server query is necessary. It is not unusual for a site to have a large number of identically configured workstations, for which all network differences derive ultimately from each workstation's unique Ethernet address.

The network interface continually monitors all arriving packets; if it sees any packet containing a destination address that matches its own physical address, it grabs the packet and forwards it to the attached CPU (via a CPU interrupt).

Ethernet also has a designated **broadcast address**. A host sending to the broadcast address has its packet received by every other host on the network; if a switch receives a broadcast packet on one port, it forwards the packet out every other port. This broadcast mechanism allows host A to contact host B when A does not yet know B's physical address; typical broadcast queries have forms such as "Will the designated server please answer" or (from the ARP protocol) "will the host with the given IP address please tell me your physical address".

Traffic addressed to a particular host – that is, not broadcast – is said to be **unicast**.

Because Ethernet addresses are assigned by the hardware, knowing an address does not provide any direct indication of where that address is located on the network. In switched Ethernet, the switches must thus have a forwarding-table record for each individual Ethernet address on the network; for extremely large networks this ultimately becomes unwieldy. Consider the analogous situation with postal addresses: Ethernet is somewhat like attempting to deliver mail using social-security numbers as addresses, where each postal worker is provided with a large catalog listing each person's SSN together with their physical location. Real postal mail is, of course, addressed "hierarchically" using ever-more-precise specifiers: state, city, zipcode, street address, and name / room#. Ethernet, in other words, does not scale well to "large" sizes.

Switched Ethernet works quite well, however, for networks with up to 10,000-100,000 nodes. Forwarding tables with size in that range are straightforward to manage.

To forward packets correctly, switches must know where all active destination addresses in the LAN are located; traditional Ethernet switches do this by a passive **learning** algorithm. (IP routers, by comparison, use "active" protocols, and some newer Ethernet switches take the approach of *2.8 Software-Defined Networking*.) Typically a host physical address is entered into a switch's forwarding table when a packet from that host is first *received*; the switch notes the packet's arrival interface and *source* address and assumes that the same interface is to be used to deliver packets back to that sender. If a given destination address has

not yet been seen, and thus is not in the forwarding table, Ethernet switches still have the backup delivery option of **flooding**: forwarding the packet to everyone by treating the destination address like the broadcast address, and allowing the host Ethernet cards to sort it out. Since this broadcast-like process is not generally used for more than one packet (after that, the switches will have learned the correct forwarding-table entries), the risks of excessive traffic and of eavesdropping are minimal.

The ⟨host,interface⟩ forwarding table is often easier to think of as ⟨host,next_hop⟩, where the next_hop node is whatever switch or host is at the immediate other end of the link connecting to the given interface. In a fully switched network where each link connects only two interfaces, the two perspectives are equivalent.

1.10 IP - Internet Protocol

To solve the scaling problem with Ethernet, and to allow support for other types of LANs and point-to-point links as well, the **Internet Protocol** was developed. Perhaps the central issue in the design of IP was to support universal connectivity (everyone can connect to everyone else) in such a way as to allow scaling to enormous size (in 2013 there appear to be around ~10^9 nodes, although IP should work to 10^{10} nodes or more), without resulting in unmanageably large forwarding tables (currently the largest tables have about 300,000 entries.)

In the early days, IP networks were considered to be "internetworks" of basic networks (LANs); nowadays users generally ignore LANs and think of the Internet as one large (virtual) network.

To support universal connectivity, IP provides a global mechanism for **addressing and routing**, so that packets can actually be delivered from any host to any other host. IP addresses (for the most-common version 4, which we denote **IPv4**) are 4 bytes (32 bits), and are part of the **IP header** that generally follows the Ethernet header. The Ethernet header only stays with a packet for one hop; the IP header stays with the packet for its entire journey across the Internet.

An essential feature of IPv4 (and IPv6) addresses is that they can be divided into a **network** part (a prefix) and a **host** part (the remainder). The "legacy" mechanism for designating the IPv4 network and host address portions was to make the division according to the first few bits:

first few bits	first byte	network bits	host bits	name	application
0	0-127	8	24	class A	a few very large networks
10	128-191	16	16	class B	institution-sized networks
110	192-223	24	8	class C	sized for smaller entities

For example, the original IP address allocation for Loyola University Chicago was 147.126.0.0, a class B. In binary, 147 is **10**010011.

IP addresses, unlike Ethernet addresses, are **administratively assigned**. Once upon a time, you would get your Class B network prefix from the Internet Assigned Numbers Authority, or IANA (they now delegate this task), and then you would in turn assign the host portion in a way that was appropriate for your local site. As a result of this administrative assignment, an IP address usually serves not just as an **endpoint identifier** but also as a **locator**, containing embedded location information (at least in the sense of location within the IP-address-assignment hierarchy, which may not be geographical). Ethernet addresses, by comparison, are endpoint identifiers but *not* locators.

3 OTHER LANS

In the wired era, one could get along quite well with nothing but Ethernet and the occasional long-haul point-to-point link joining different sites. However, there are important alternatives out there. Some, like token ring, are mostly of historical importance; others, like virtual circuits, are of great conceptual importance but – so far – of only modest day-to-day significance.

And then there is wireless. It would be difficult to imagine contemporary laptop networking, let alone mobile devices, without it. In both homes and offices, Wi-Fi connectivity is the norm. Mobile networking is ubiquitous. A return to being tethered by wires is almost unthinkable.

3.1 Virtual Private Networks

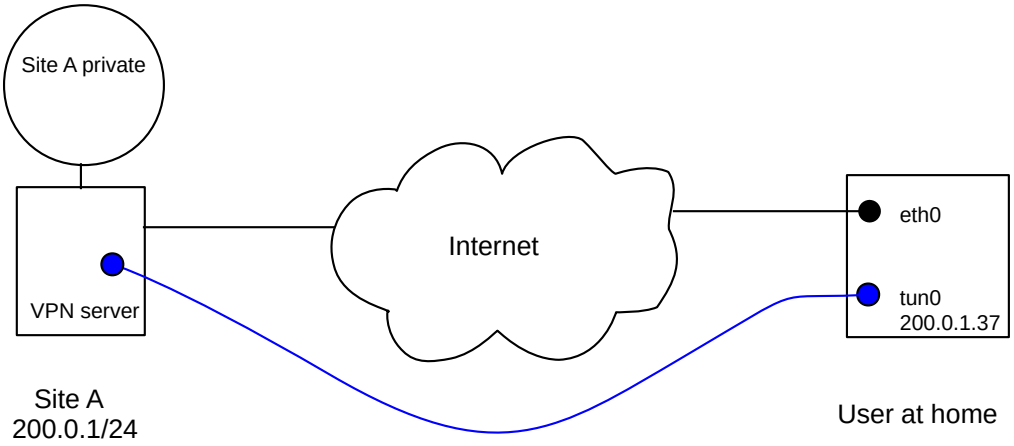
Suppose you want to connect to your workplace network from home. Your workplace, however, has a security policy that does not allow "outside" IP addresses to access essential internal resources. How do you proceed, without leasing a dedicated telecommunications line to your workplace?

A virtual private network, or **VPN**, provides a solution; it supports creation of **virtual links** that join far-flung nodes via the Internet. Your home computer creates an ordinary Internet connection (TCP or UDP) to a workplace **VPN server** (IP-layer packet encapsulation can also be used, and avoids the timeout problems sometimes created by sending TCP packets within another TCP stream; see *7.13 Mobile IP*). Each end of the connection is typically associated with a software-created **virtual network interface**; each of the two virtual interfaces is assigned an IP address. (Virtual interfaces are not essential; VPNs created with IPsec, *22.11 IPsec*, generally omit them.) When a packet is to be sent along the virtual link, it is actually encapsulated and sent along the original Internet connection to the VPN server, wending its way through the commodity Internet; this process is called **tunneling**. To all intents and purposes, the virtual link behaves like any other physical link.

Tunneled packets are often encrypted as well as encapsulated, though that is a separate issue. One relatively easy-to-implement example of a tunneling mechanism is to treat a TCP home-workplace connection as a serial line and send packets over it back-to-back, using PPP with HDLC; see *4.1.5.1 HDLC* and **RFC 1661** (though this can lead to the above-mentioned TCP-in-TCP timeout problems).

At the workplace side, the virtual network interface in the VPN server is attached to a router or switch; at the home user's end, the virtual network interface can now be assigned an *internal* workplace IP address. The home computer is now, for all intents and purposes, part of the internal workplace network.

In the diagram below, the user's regular Internet connection is via hardware interface `eth0`. A connection is established to Site A's VPN server; a virtual interface `tun0` is created on the user's machine which appears to be a direct link to the VPN server. The `tun0` interface is assigned a Site-A IP address. Packets sent via the `tun0` interface in fact travel over the original connection via `eth0` and the Internet.



VPN: blue link represents *tunnel.* Actual connection is made via eth0
The tun0 interface is a virtual network interface with a Site-A address

After the VPN is set up, the home host's `tun0` interface appears to be locally connected to Site A, and thus the home host is allowed to connect to the private area within Site A. The home host's forwarding table will be configured so that traffic to Site A's private addresses is routed via interface `tun0`.

VPNs are also commonly used to connect entire remote offices to headquarters. In this case the remote-office end of the tunnel will be at that office's local router, and the tunnel will carry traffic for all the workstations in the remote office.

Other applications of VPNs include trying to appear geographically to be at another location, and bypassing firewall rules blocking specific TCP or UDP ports.

To improve security, it is common for the residential (or remote-office) end of the VPN connection to use the VPN connection as the default route for all traffic *except* that needed to maintain the VPN itself. This may require a so-called **host-specific** forwarding-table entry at the residential end to allow the packets that carry the VPN tunnel traffic to be routed correctly via `eth0`. This routing strategy means that potential intruders cannot access the residential host – and thus the workplace internal network – through the original residential Internet access. A consequence is that if the home worker downloads a large file from a non-workplace site, it will travel first to the workplace, then back out to the Internet via the VPN connection, and finally arrive at the home.

To improve congestion response, IP packets are sometimes marked by routers that are experiencing congestion; see *14.8.3 Explicit Congestion Notification (ECN)*. If such marking is done to the outer, encapsulating, packet, and the marks are not transferred at the remote endpoint of the VPN to the inner, encapsulat*ed*, packet, then the marks are lost. Congestion response may suffer. **RFC 6040** spells out a proper re-marking strategy in general; **RFC 7296** defines re-marking for IPsec (*22.11 IPsec*). Older VPN protocols, however, may not support congestion re-marking.

3.2 Carrier Ethernet

Carrier Ethernet is a leased-line point-to-point link between two sites, where the subscriber interface at each end of the line looks like Ethernet (in some flavor). The physical path in between sites, however, need not

have anything to do with Ethernet; it may be implemented however the carrier wishes. In particular, it will be (or at least appear to be) full-duplex, it will be collision-free, and its length may far exceed the maximum permitted by any IEEE Ethernet standard.

Bandwidth can be purchased in whatever increments the carrier has implemented. The point of carrier Ethernet is to provide a layer of abstraction between the customers, who need only install a commodity Ethernet interface, and the provider, who can upgrade the link implementation at will without requiring change at the customer end.

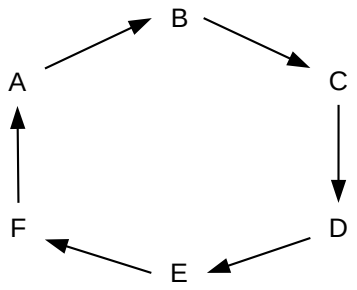
In a sense, carrier Ethernet is similar to the widespread practice of provisioning residential DSL and cable routers with an Ethernet interface for customer interconnection; again, the actual link technologies may not look anything like Ethernet, but the interface will.

A carrier Ethernet connection looks like a virtual VPN link, but runs on top of the provider's internal network rather than the Internet at large. Carrier Ethernet connections often provide the primary Internet connectivity for one endpoint, unlike Internet VPNs which assume both endpoints already have full Internet connectivity.

3.3 Token Ring

A significant part of the previous chapter was devoted to classic Ethernet's collision mechanism for supporting shared media access. After that, it may come as a surprise that there is a simple multiple-access mechanism that is not only **collision-free**, but which supports **fairness** in the sense that if N stations wish to send then each will receive 1/N of the opportunities.

That method is **Token Ring**. Actual implementations come in several forms, from Fiber-Distributed Data Interface (FDDI) to so-called "IBM Token Ring". The central idea is that stations are connected in a ring:



Packets will be transmitted in one direction (clockwise in the ring above). Stations in effect forward most packets around the ring, although they can also remove a packet. (It is perhaps more accurate to think of the forwarding as representing the default cable connectivity; *non*-forwarding represents the station's momentarily breaking that connectivity.)

When the network is idle, all stations agree to forward a special, small packet known as a *token*. When a station, say A, wishes to transmit, it must first wait for the token to arrive at A. Instead of forwarding the token, A then transmits its own packet; this travels around the network and is then removed by A. At that point (or in some cases at the point when A finishes transmitting its data packet) A then forwards the token.

In a small ring network, the ring circumference may be a small fraction of one packet. Ring networks become "large" at the point when some packets may be entirely in transit on the ring. Slightly different

solutions apply in each case. (It is also possible that the physical ring exists only within the token-ring switch, and that stations are connected to that switch using the usual point-to-point wiring.)

If all stations have packets to send, then we will have something like the following:

- A waits for the token
- A sends a packet
- A sends the token to B
- B sends a packet
- B sends the token to C
- C sends a packet
- C sends the token to D
- ...

All stations get an equal number of chances to transmit, and no bandwidth is wasted on collisions. (A station constantly sending smaller packets will send the same number of packets as a station constantly sending larger packets, but the bandwidth will be smaller in proportion to the smaller packet size.)

One problem with token ring is that when stations are powered off it is *essential* that the packets continue forwarding; this is usually addressed by having the default circuit configuration be to keep the loop closed. Another issue is that some station has to watch out in case the token disappears, or in case a duplicate token appears.

Because of fairness and the lack of collisions, IBM Token Ring was once considered to be the premium LAN mechanism. As such, Token Ring hardware commanded a substantial price premium. But due to Ethernet's combination of lower hardware costs and higher bitrates (even taking collisions into account), the latter eventually won out.

There was also a much earlier collision-free hybrid of 10 Mbps Ethernet and Token Ring known as **Token Bus**: an Ethernet physical network (often linear) was used with a token-ring-like protocol layer above that. Stations were physically connected to the (linear) Ethernet but were assigned identifiers that logically arranged them in a (virtual) ring. Each station had to wait for the token and only then could transmit a packet; after that it would send the token on to the next station in the virtual ring. As with "real" Token Ring, some mechanisms need to be in place to monitor for token loss.

Token Bus Ethernet never caught on. The additional software complexity was no doubt part of the problem, but perhaps the real issue was that it was not necessary.

3.4 Virtual Circuits

Before we can get to our final LAN example, ATM, we need to detour briefly through virtual circuits.

The Road Not Taken

A close reading of Robert Frost's poem referenced here reveals that the supposed great difference between the two roads exists only in the narrator's retrospective imaginings; the roads were in fact "really about the same". Perhaps this would also apply to datagram and virtual-circuit forwarding, though see below on per-connection billing.

Virtual circuits are The Road Not Taken by IP.

Virtual-circuit switching (or routing) is an alternative to datagram switching, which was introduced in Chapter 1. In datagram switching, routers know the next_hop to each destination, and packets are addressed by *destination*. In virtual-circuit switching, routers know about end-to-end *connections*, and packets are "addressed" by a connection ID.

Before any data packets can be sent, a connection needs to be established first. For that connection, the route is computed and then each link along the path is assigned a connection ID, traditionally called the **VCI**, for Virtual Circuit Identifier. In most cases, VCIs are only *locally* unique; that is, the same connection may use a different VCI on each link. The lack of global uniqueness makes VCI allocation much simpler. Although the VCI keeps changing along a path, the VCI can still be thought of as identifying the connection. To send a packet, the host marks the packet with the VCI assigned to the host–router1 link.

Packets arrive at (and depart from) switches via one of several **ports**, which we will assume are numbered beginning at 0. Switches maintain a **connection table** indexed by $\langle$VCI,port$\rangle$ pairs; unlike a forwarding table, the connection table has a record of every connection through that switch at that particular moment. As a packet arrives, its inbound VCI_{in} and inbound port_{in} are looked up in this table; this yields an outbound $\langle \text{VCI}_{\text{out}}, \text{port}_{\text{out}} \rangle$ pair. The VCI field of the packet is then *rewritten* to VCI_{out}, and the packet is sent via port_{out}.

Note that typically there is no source address information included in the packet (although the sender can be identified from the connection, which can be identified from the VCI at any point along the connection). Packets are identified by connection, not destination. Any node along the path (including the endpoints) can in principle look up the connection and figure out the endpoints.

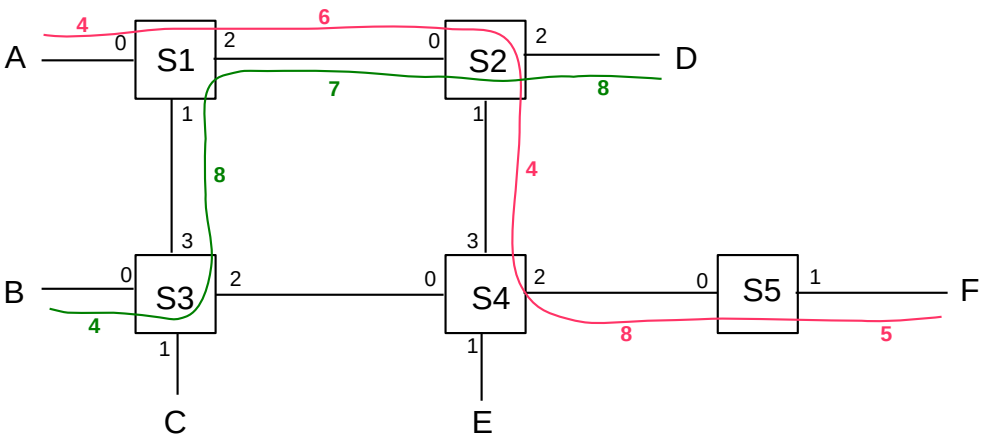
Note also that each switch must rewrite the VCI. Datagram switches never rewrite addresses (though they do update hopcount/TTL fields). The advantage to this rewriting is that VCIs need be unique only for a given link, greatly simplifying the naming. Datagram switches also do not make use of a packet's arrival interface.

As an example, consider the network below. Switch ports are numbered 0,1,2,3. Two paths are drawn in, one from A to F in red and one from B to D in green; each link is labeled with its VCI number in the same color.

We will construct virtual-circuit connections between

- A and F (shown above in red)
- A and E
- A and C
- B and D (shown above in green)
- A and F again (a separate connection)

The following VCIs have been chosen for these connections. The choices are made more or less randomly here, but in accordance with the requirement that they be unique to each link. Because links are generally

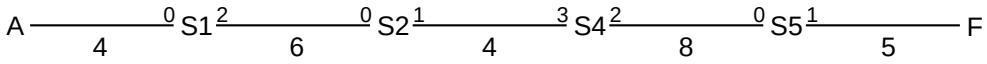


taken to be bidirectional, a VCI used from S1 to S3 cannot be reused from S3 to S1 until the first connection closes.

- A to F: A—4—S1—6—S2—4—S4—8—S5—5—F; this path goes from S1 to S4 via S2
- A to E: A—5—S1—6—S3—3—S4—8—E; this path goes, for no particular reason, from S1 to S4 via S3, the opposite corner of the square
- A to C: A—6—S1—7—S3—3—C
- B to D: B—4—S3—8—S1—7—S2—8—D
- A to F: A—7—S1—8—S2—5—S4—9—S5—2—F

One may verify that on any one link no two different paths use the same VCI.

We now construct the actual ⟨VCI,port⟩ tables for the switches S1-S4, from the above; the table for S5 is left as an exercise. Note that either the $\langle VCI_{in}, port_{in} \rangle$ or the $\langle VCI_{out}, port_{out} \rangle$ can be used as the key; we cannot have the same pair in both the in columns and the out columns. It may help to display the port numbers for each switch, as in the upper numbers in following diagram of the above red connection from A to F (lower numbers are the VCIs):



Switch **S1**:

VCI_{in}	$port_{in}$	VCI_{out}	$port_{out}$	connection
4	0	6	2	A⟶F #1
5	0	6	1	A⟶E
6	0	7	1	A⟶C
8	1	7	2	B⟶D
7	0	8	2	A⟶F #2

Switch **S2**:

VCI_{in}	$port_{in}$	VCI_{out}	$port_{out}$	connection
6	0	4	1	A⟶F #1
7	0	8	2	B⟶D
8	0	5	1	A⟶F #2

Switch **S3**:

VCI_{in}	$port_{in}$	VCI_{out}	$port_{out}$	connection
6	3	3	2	A⟶E
7	3	3	1	A⟶C
4	0	8	3	B⟶D

Switch **S4**:

VCI_{in}	$port_{in}$	VCI_{out}	$port_{out}$	connection
4	3	8	2	A⟶F #1
3	0	8	1	A⟶E
5	3	9	2	A⟶F #2

The namespace for VCIs is small, and compact (*eg* contiguous). Typically the VCI and port bitfields can be concatenated to produce a ⟨VCI,Port⟩ composite value small enough that it is suitable for use as an array index. VCIs work best as *local* identifiers. IP addresses, on the other hand, need to be globally unique, and thus are often rather sparsely distributed.

Virtual-circuit switching offers the following advantages:

- connections can get quality-of-service guarantees, because the switches are aware of connections and can reserve capacity at the time the connection is made
- headers are smaller, allowing faster throughput
- headers are small enough to allow efficient support for the very small packet sizes that are optimal for voice connections. ATM packets, for instance, have 48 bytes of data; see below.

Datagram forwarding, on the other hand, offers these advantages:

- Routers have less state information to manage.
- Router crashes and partial connection state loss are not a problem.
- If a router or link is disabled, rerouting is easy and does not affect any connection state. (As mentioned in Chapter 1, this was Paul Baran's primary concern in his 1962 paper introducing packet switching.)
- Per-connection billing is very difficult.

The last point above may once have been quite important; in the era when the ARPANET was being developed, typical daytime long-distance rates were on the order of $1/minute. It is unlikely that early TCP/IP protocol development would have been as fertile as it was had participants needed to justify per-minute billing costs for every project.

It is certainly possible to do virtual-circuit switching with globally unique VCIs – say the concatenation of source and destination IP addresses and port numbers. The IP-based RSVP protocol (*20.6 RSVP*) does exactly this. However, the fast-lookup and small-header advantages of a compact namespace are then lost.

Multi-Protocol Label Switching (*20.12 Multi-Protocol Label Switching (MPLS)*) is another IP-based application of virtual circuits.

Note that virtual-circuit switching does *not* suffer from the problem of idle channels still consuming resources, which is an issue with circuits using time-division multiplexing (*eg* shared T1 lines)

3.5 Asynchronous Transfer Mode: ATM

ATM is a network mechanism intended to accommodate real-time traffic as well as bulk data transfer. We present ATM here as a LAN layer, for which it is still sometimes used, but it was originally proposed as a replacement for the IP layer as well, and, to an extent, the Transport layer. These broader plans were not greeted with universal enthusiasm within the IETF. When used as a LAN layer, IP packets are transmitted over ATM as in *3.5.1 ATM Segmentation and Reassembly*.

A distinctive feature of ATM is its small packet size. ATM has its roots in the telephone industry, and was therefore particularly intended to support voice. A significant source of delay in voice traffic is the packet **fill time**: at DS0 speeds (64 kbps), voice data accumulates at 8 bytes/ms. If we are sending 1 kB packets, this means voice is delayed by about 1/8 second, meaning in turn that when one person stops speaking, the earliest they can hear the other's response is 1/4 second later. Slightly smaller levels of voice delay can introduce an annoying echo. Smaller packets reduce the fill time and thus the delay: when voice is sent over IP (VoIP), one common method is to send 160 bytes every 20 ms.

ATM took this small-packet strategy even further: packets have 48 bytes of data, plus 5 bytes of header. Such small packets are often called *cells*. To manage such a small header, virtual-circuit routing is a necessity. IP packets of such small size would likely consume more than 50% of the bandwidth on headers, if the LAN header were included.

Aside from reduced voice fill-time, other benefits to small cells are reduced store-and-forward delay and minimal queuing delay, at least for high-priority traffic. Prioritizing traffic and giving precedence to high-priority traffic is standard, but high-priority traffic is never allowed to *interrupt* transmission already begun of a low-priority packet. If you have a high-priority voice cell, and someone else has a 1500-byte packet just started, your cell has to wait about 30 cell times, because 1500 bytes is about 30 cells. However, if their low-priority traffic is instead made up of 30 cells, you have only to wait for their first cell to finish; the delay is 1/30 as much.

ATM also made the decision to require **fixed-size** cells. The penalty for one partially used cell among many is small. Having a fixed cell size simplifies hardware design, and, in theory, allows it easier to design for parallelism.

Unfortunately, the designers of ATM also chose to mandate **no cell reordering**. This means cells can use a smaller sequence-number field, but also makes parallel switches much harder to build. A typical parallel switch design might involve distributing incoming cells among any of several input queues; the queues would then handle the VCI lookups in parallel and forward the cells to the appropriate output queues. With such an architecture, avoiding reordering is difficult. It is not clear to what extent the no-reordering decision was related to the later decline of ATM in the marketplace.

ATM cells have 48 bytes of data and a 5-byte header. The header contains up to 28 bits of VCI information, three "type" bits, one **cell-loss priority**, or CLP, bit, and an 8-bit checksum over the header only. The VCI is divided into 8-12 bits of Virtual Path Identifier and 16 bits of Virtual Channel Identifier, the latter supposedly for customer use to separate out multiple connections between two endpoints. Forwarding is by full switching only, and there is no mechanism for physical (LAN) broadcast.

3.5.1 ATM Segmentation and Reassembly

Due to the small packet size, ATM defines its own mechanisms for segmentation and reassembly of larger packets. Thus, individual ATM links in an IP network are quite practical. These mechanisms are called **ATM Adaptation Layers**, and there are four of them: AALs 1, 2, 3/4 and 5 (AAL 3 and AAL 4 were once separate layers, which merged). AALs 1 and 2 are used only for voice-type traffic; we will not consider them further.

The ATM segmentation-and-reassembly mechanism defined here is intended to apply only to large *data*; no cells are ever further subdivided. Furthermore, segmentation is always applied at the point where the data enters the network; reassembly is done at exit from the ATM path. IPv4 fragmentation, on the other hand, applies conceptually to IP packets, and may be performed by routers within the network.

For AAL 3/4, we first define a high-level "wrapper" for an IP packet, called the CS-PDU (Convergence Sublayer - Protocol Data Unit). This prefixes 32 bits on the front and another 32 bits (plus padding) on the rear. We then chop this into as many 44-byte chunks as are needed; each chunk goes into a 48-byte ATM payload, along with the following 32 bits worth of additional header/trailer:

- 2-bit **type** field:
  - 10: begin new CS-PDU
  - 00: continue CS-PDU
  - 01: end of CS-PDU
  - 11: single-segment CS-PDU
- 4-bit sequence number, 0-15, good for catching up to 15 dropped cells
- 10-bit MessageID field
- CRC-10 checksum.

We now have a total of 9 bytes of header for 44 bytes of data; this is more than 20% overhead. This did not sit well with the IP-over-ATM community (such as it was), and so AAL 5 was developed.

AAL 5 moved the checksum to the CS-PDU and increased it to 32 bits from 10 bits. The MID field was discarded, as no one used it, anyway (if you wanted to send several different types of messages, you simply created several virtual circuits). A bit from the ATM header was taken over and used to indicate:

- 1: start of new CS-PDU
- 0: continuation of an existing CS-PDU

The CS-PDU is now chopped into 48-byte chunks, which are then used as the entire body of each ATM cell. With 5 bytes of header for 48 bytes of data, overhead is down to 10%. Errors are detected by the

CS-PDU CRC-32. This also detects lost cells (impossible with a per-cell CRC!), as we no longer have any cell sequence number.

For both AAL3/4 and AAL5, **reassembly** is simply a matter of stringing together consecutive cells **in order of arrival**, starting a new CS-PDU whenever the appropriate bits indicate this. For AAL3/4 the receiver has to strip off the 4-byte AAL3/4 headers; for AAL5 the receiver has to verify the CRC-32 checksum once all cells are received. Different cells from different virtual circuits can be jumbled together on the ATM "backbone", but on any one virtual circuit the cells from one higher-level packet must be sent one right after the other.

A typical IP packet divides into about 20 cells. For AAL 3/4, this means a total of 200 bits devoted to CRC codes, versus only 32 bits for AAL 5. It might seem that AAL 3/4 would be more reliable because of this, but, paradoxically, it was not! The reason for this is that errors are *rare*, and so we typically have one or at most two per CS-PDU. Suppose we have only a single error, *ie* a single cluster of corrupted bits small enough that it is likely confined to a single cell. In AAL 3/4 the CRC-10 checksum will fail to detect that error (that is, the checksum of the corrupted packet will by chance happen to equal the checksum of the original packet) with probability $1/2^{10}$. The AAL 5 CRC-32 checksum, however, will fail to detect the error with probability $1/2^{32}$. Even if there are enough errors that two cells are corrupted, the two CRC-10s together will fail to detect the error with probability $1/2^{20}$; the CRC-32 is better. AAL 3/4 is more reliable only when we have errors in at least four cells, at which point we might do better to switch to an error-*correcting* code.

Moral: one checksum over the entire message is often better than multiple shorter checksums over parts of the message.

3.6 Adventures in Radioland

For the remainder of this chapter we leave wires (and fiber) behind, and contemplate the transmission of packets via radio, freeing nodes from their cable tethers. Wi-fi (*3.7 Wi-Fi*) and mobile wireless (*3.8 WiMAX and LTE*) are now ubiquitous. But radio is not quite like wire, and wireless transmission of packets brings several changes.

3.6.1 Privacy

It's hard to tap into wired Ethernet, especially if you are locked out of the building. But anyone can receive wireless transmissions, often from a considerable distance. The data breach at TJX Corporation was achieved by attackers parking outside a company building and pointing a directional antenna at it; encryption was used but it was weak (see *22 Security* and *22.7.7 Wi-Fi WEP Encryption Failure*). Similarly, Internet café visitors generally don't want other patrons to read their email. Radio communication needs strong encryption.

3.6.2 Collisions

Ethernet-like **collision detection** is no longer feasible over radio. This has to do with the relative signal strength of the remote signal at the local transmitter. Along a wire-based Ethernet the remote signal might be as weak as 1/100 of the transmitted signal but that 1% received signal is still detectable *during* transmission.

However, with radio the remote signal might easily be as little as 1/1,000,000 of the transmitted signal (-60 dB), as measured at the transmitting station, and it is simply overwhelmed during transmission.

As a result, wireless protocols must be constructed appropriately. We will look at how Wi-Fi handles this in its most common mode of operation in *3.7.1 Wi-Fi and Collisions*. Wi-Fi also supports its PCF mode (*3.7.7 Wi-Fi Polling Mode*) that involves fewer (but not zero) collisions through the use of central-point **polling**. Finally, WiMAX and LTE switch from polling to **scheduling** to further reduce collisions, though the potential for collisions is still inevitable when new stations join the network.

It is also worth pointing out that, while an Ethernet collision affects every station in the physical Ethernet (the "collision domain"), wireless collisions are **local**, occuring at the receiver. Two stations can transmit at the same time, and in range of one another, but without a collision! This can happen if each of the relevant *receivers* is in range of only one of the two transmitting stations. As an example, suppose three stations are arranged linearly, A–C–B, with the A–C and C–B distances just under the maximum effective range. When A and B both transmit there is indeed a collision at C. But when C and B transmit simultaneously, A may receive C's signal just fine, as B's is too weak to interfere.

3.6.3 Hidden Nodes

In wireless communication, two nodes A and B that are not in range of one another – and thus cannot detect one another – may still have their signals interfere at a third node C. This creates an additional complication to collision handling. See *3.7.1.4 Hidden-Node Problem*.

3.6.4 Band Width

To radio engineers, "band width" means the frequency range used by a signal, not the data transmission rate. No information can be conveyed using a single frequency; even signaling by switching a carrier frequency off and on at a low rate "blurs" the carrier into a band of nonzero width.

In keeping with this we will for the remainder of this chapter use the term "data rate" for what we have previously called "bandwidth". We will use the terms "channel width" or "width of the frequency band" for the frequency range.

All else being equal, the data rate achievable with a radio signal is proportional to the channel width. The constant of proportionality is limited by the Shannon-Hartley theorem: the maximum data rate divided by the width of the frequency band is $\log_2(1+\text{SNR})$, where SNR is the **signal to noise** power ratio. Noise here is assumed to have a specific statistical form known as *Gaussian white noise*. If SNR is 127, for example, and the width of the frequency band is 1 MHz, then the maximum theoretical data rate is 7 Mbps, where $7 = \log_2(128)$. If the signal power S drops by about half so SNR=63, the data rate falls to 6 Mbps, as $6 = \log_2(64)$; the relationship between signal power and data rate is logarithmic.

3.6.4.1 OFDM

The actual data rate achievable, for a given channel width and SNR, depends on the signal encoding, or **modulation**, mechanism. Most newer modulation mechanisms use "orthogonal frequency-division multiplexing", **OFDM**, or some variant.

A central feature of OFDM is that one wider frequency band is divided into multiple narrow subchannels; each subchannel then carries a proportional fraction of the total information signal, modulated onto a subchannel-specific carrier. All the subchannels can be allocated to one transmission at a time (time-division multiplexing, *4.2 Time-Division Multiplexing*), or disjoint sets of subchannels can be allocated to different transmissions that can then proceed (at proportionally lower data rates) in parallel. The latter is known as *frequency-division multiplexing*.

In many settings OFDM comes reasonably close to the Shannon-Hartley limit. Perhaps more importantly, OFDM also performs reasonably well with *multipath interference*, below, which is endemic in urban and building-interior environments with their many reflective surfaces. Multipath interference is, however, not necessarily comparable to the Gaussian noise assumed by the Shannon-Hartley theorem. We will not address further technical details of OFDM here, except to note that implementation usually requires some form of digital signal processing.

The OFDMA variant, with the MA standing for Multiple Access, allows multiple users to use nonoverlapping sets of subchannels, thus allowing simultaneous transmission. It is an option available in 802.11ax.

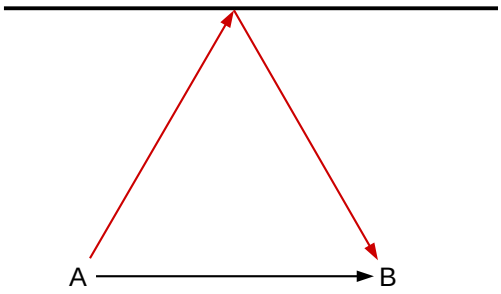
3.6.5 Cost

Another fundamental issue is that everyone shares the same radio spectrum. For mobile wireless providers, this constraint has driven prices to remarkable levels; the 2014-15 FCC AWS-3 auction raised almost $45 billion for 65 MHz (usable throughout the entire United States). This works out to somewhat over $2 per megahertz per phone. The corresponding issue for Wi-Fi users in a dense area is that all the available Wi-Fi bandwidth may be in use. Inside busy buildings one can often see dozens of Wi-Fi access points competing for the same Wi-Fi channel; the result is that no user will be getting close to the nominal data rates of *3.7 Wi-Fi*.

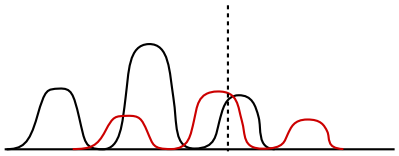
Higher data rates require wider frequency bands. To reduce costs in the face of fixed demand, the usual strategy is to make the coverage zones smaller, either by reducing power (and adding more access points as appropriate), or by using directional antennas, or both.

3.6.6 Multipath

While a radio signal generally covers a wide area – even with ordinary directional antennas – it does so in surprisingly non-uniform ways. A signal may reach a receiver through a line-of-sight path and also several reflected paths, possibly of varying length. In addition to reflection, the signal may be subject to reflection-like *scattering* and *diffraction*. All of this together is known as **multipath interference** (or, if analog audio is involved, multipath *distortion*; in the analog TV era this was *ghosting*).



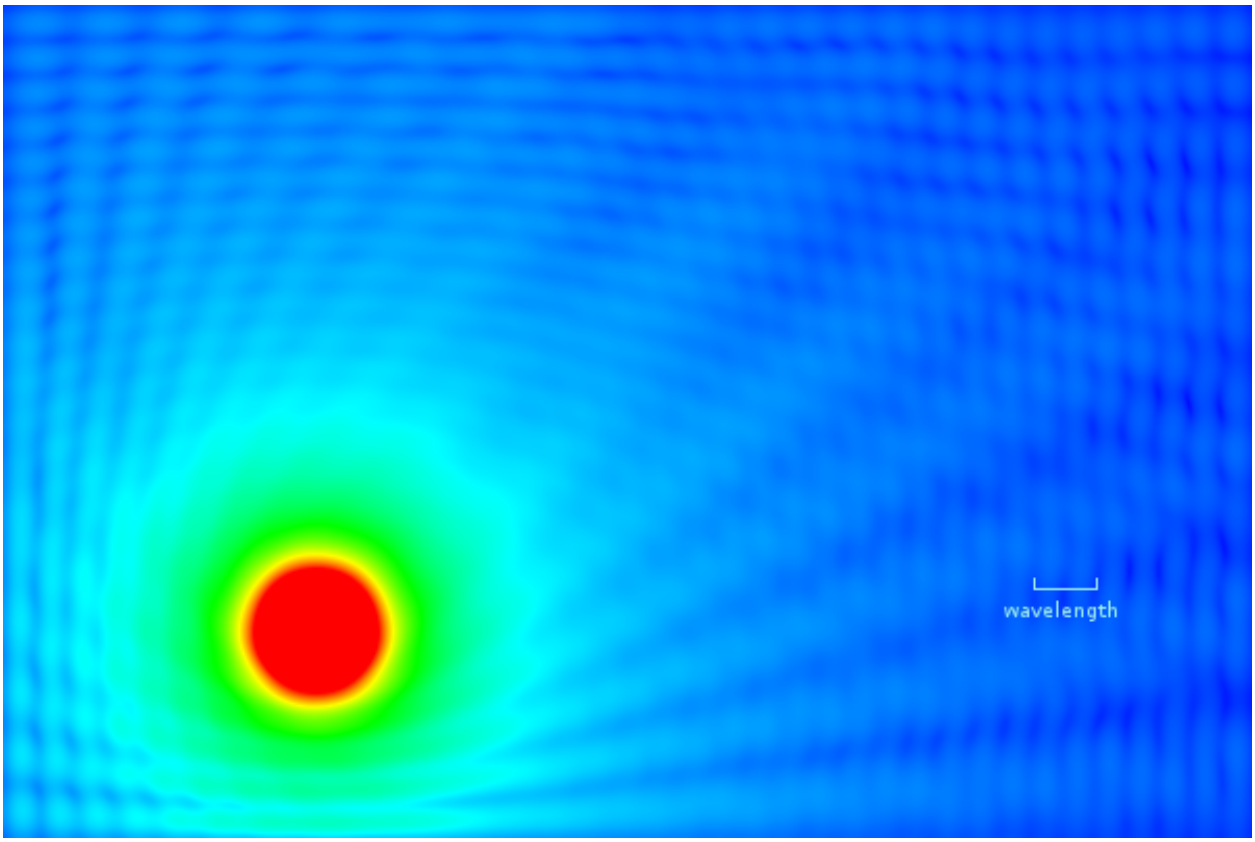
Line-of-sight and reflected signals



Superposition of encoded data

The picture above shows two transmission paths from A to B. The respective carrier paths may interfere with or supplement one another. The longer delay of the reflecting path (red) wil also delay its encoded signal. The result, shown at right, is that the line-of-sight and reflected data symbols may overlap and interfere with each other; this is known as **intersymbol interference**. Multipath interference may even change the meaning of the data symbol as seen by the receiver; for example, the red and black low data-signal peaks above at the point of the vertical dashed line may sum together so as to be received as a higher peak (assuming the underlying carriers are in sync).

Multipath interference tends to lead to wide fluctuations in signal intensity with a period of about half a wavelength;



Signal-intensity map (simulated) in a room with walls with 40% reflectivity

The picture above is from a mathematical simulation intended to illustrate multipath fading. The walls of the room reflect 40% of the signal from the transmitter located in the orange ball at the lower left. The transmitter transmits an unmodulated carrier signal, which may be reflected off the walls any number of times; at any point in the room the total signal intensity is the sum over all possible reflection paths. On the right-hand side, the small-scale blue ripples represent the received carrier strength variation due to multipath interference between the line-of-sight and all the reflected paths. Note that the ripple size is about half a wavelength.

In comparison to this simulated intensity map, real walls tend to have a lower reflectivity, real rooms are not two-dimensional, and real carriers are modulated. However, real rooms also introduce scattering, diffraction and shadowing from objects within, and significant (3× to 10×) multipath-fading signal-strength variations are common in actual wireless settings.

Multipath fading can be either **flat** – affecting all frequencies more or less equally – or **selective** – affecting some frequencies differently than others. It is quite possible for an OFDM channel (*3.6.4.1 OFDM*) to encounter selective fading of only some of its subchannel frequencies.

Generally, multipath interference is a problem that engineers go to great lengths to overcome. However, as we shall see in *3.7.3 Multiple Spatial Streams*, multipath interference can sometimes be put to positive use by allowing almost-adjacent antennas to transmit and receive independent signals, thus increasing the effective throughput.

For an alternative example of multipath interference in which the signal strength has no ripples, see exercise 13.0.

3.6.7 Power

If you are cutting the network cable and replacing it with wireless, there is a good chance you will also want to cut the power cable as well and replace it with batteries. This tends to make power consumption a very important issue. The Wi-Fi standard has provisions for minimizing power usage by allowing a device to "doze" most of the time, waking periodically to check if any packets are ready to be sent to it (see *3.7.4.1 Joining a Network*). The 6LoWPAN project (IPv6 Low-power Wireless Personal Area Network) is intended to support very low-power devices; see **RFC 4919** and **RFC 6282**.

3.6.8 Tangle

Wireless is also used simply to replace cords and their attendant tangle, and, of course, the problem of incompatible connectors. The low-power **Bluetooth** wireless standard is commonly used as a cable alternative for things like computer mice and telephone headsets. Bluetooth is also a low-power network; for many applications the working range is about 10 meters. **ZigBee** is another low-power small-scale network.

3.7 Wi-Fi

Wi-Fi is a trademark of the Wi-Fi Alliance denoting any of several IEEE wireless-networking protocols in the 802.11 family, specifically 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac and 802.11ax. (Strictly

speaking, these are all *amendments* to the original 802.11 standard, but they are also *de facto* standards in their own right.) Like classic Ethernet, Wi-Fi must deal with **collisions**; unlike Ethernet, however, Wi-Fi is unable to detect collisions in progress, complicating the backoff and retransmission algorithms. See *3.6.2 Collisions* above.

Unlike any wired LAN protocol we have considered so far, in addition to normal data packets Wi-Fi also uses **control** and **management** packets that exist entirely within the Wi-Fi LAN layer; these are not initiated by or delivered to higher network layers. Control packets are used to compensate for some of the infelicities of the radio environment, such as the lack of collision detection. Putting radio-essential control and management protocols within the Wi-Fi layer means that the IP layer can continue to interact with the Wi-Fi LAN exactly as it did with Ethernet; no changes are required.

Wi-Fi is designed to interoperate freely with Ethernet at the logical LAN layer. Wi-Fi MAC (physical) addresses have the same 48-bit size as Ethernet's and the same internal structure (*2.1.3 Ethernet Address Internal Structure*). They also share the same namespace: one is never supposed to see an Ethernet and a Wi-Fi interface with the same address. As a result, data packets can be forwarded by switches between Ethernet and Wi-Fi; in many respects a Wi-Fi LAN attached to an Ethernet LAN looks like an extension of the Ethernet LAN. See *3.7.4 Access Points*.

Microwave Ovens and Wi-Fi

The impact of a running microwave oven on Wi-Fi signals is quite evident if the oven is between the sender and receiver. For other configurations the effect may vary. Most ovens transmit only during one half of the A/C cycle, that is, they are on 1/60 sec and then off 1/60 sec; this may allow intervening transmission time. See also here.

Traditionally, Wi-Fi used the 2.4 GHz **ISM** (Industrial, Scientific and Medical) band used also by microwave ovens, though 802.11a used a 5 GHz band, 802.11n supports that as an option and the new 802.11ac has returned to using 5 GHz exclusively. The 5 GHz band has reduced ability to penetrate walls, often resulting in a lower effective range (though in offices and multi-unit housing this can be an advantage). The 5 GHz band provides many more usable channels than the 2.4 GHz band, resulting in much less interference in "crowded" environments.

Wi-Fi radio spectrum is usually **unlicensed**, meaning that no special permission is needed to transmit but also that others may be trying to use the same frequency band simultaneously. The availability of unlicensed channels in the 5 GHz band continues to improve.

The table below summarizes the different Wi-Fi versions. All data bit rates assume a single spatial stream; channel widths are nominal. The names in the far-right column have been introduced by the Wi-Fi Alliance as a more convenient designation for the newer versions.

IEEE name	maximum bit rate	frequency	channel width	new name
802.11a	54 Mbps	5 GHz	20 MHz	
802.11b	11 Mbps	2.4 GHz	20 MHz	
802.11g	54 Mbps	2.4 GHz	20 MHz	
802.11n	65-150 Mbps	2.4/5 GHz	20-40 MHz	Wi-Fi 4
802.11ac	78-867 Mbps	5 GHz	20-160 MHz	Wi-Fi 5
802.11ax	Up to 1200 Mbps	2.4/5+ GHz	20-160 MHz	Wi-Fi 6

The maximum bit rate is seldom achieved in practice. The *effective* bit rate must take into account, at a minimum, the time spent in the collision-handling mechanism. More significantly, all the Wi-Fi variants above use dynamic rate scaling, below; the bit rate is reduced up to tenfold (or more) in environments with higher error rates, which can be due to distance, obstructions, competing transmissions or radio noise. All this means that, as a practical matter, getting 150 Mbps out of 802.11n requires optimum circumstances; in particular, no competing senders and unimpeded line-of-sight transmission. 802.11n lower-end performance can be as little as 10 Mbps, though 40-100 Mbps (for a 40 MHz channel) may be more typical.

The 2.4 GHz ISM band is divided by international agreement into up to 14 officially designated (and mostly adjacent) channels, each about 5 MHz wide, though in the United States use may be limited to the first 11 channels. The 5 GHz band is similarly divided into 5 MHz channels. One Wi-Fi sender, however, needs several of these official channels; the typical 2.4 GHz 802.11g transmitter uses an actual frequency range of up to 22 MHz, or up to five official channels. As a result, to avoid signal overlap Wi-Fi use in the 2.4 GHz band is often restricted to official channels 1, 6 and 11. The end result is that there are generally only three available Wi-Fi bands in the 2.4 GHz range, and so Wi-Fi transmitters can and do interact with and interfere with each other.

There are almost 200 5 MHz channels in the 5 GHz band. The United States requires users of the this band to avoid interfering with weather and military applications in the same frequency range; this may involve careful control of transmission power (under the IEEE 802.11h amendment) and so-called "dynamic frequency selection" to choose channels with little interference, and to switch to such channels if interference is detected later. Even so, there are many more channels than at 2.4 GHz; the larger number of channels is one of the reasons (arguably the primary reason) that 802.11ac can run faster (below). The number of channels available for Wi-Fi use has been increasing, often as conflicts with existing 5 GHz weather systems are resolved.

802.11ax has preliminary support for additional frequency bands in the 6-7 GHz range, though these are still awaiting (in the US) final FCC approval.

Wi-Fi designers can improve throughput through a variety of techniques, including

1. improved radio modulation techniques
2. improved error-correcting codes
3. smaller guard intervals between symbols
4. increasing the channel width
5. allowing multiple spatial streams via multiple antennas

The first two in this list seem now to be largely tapped out; OFDM modulation (*3.6.4.1 OFDM*) is close enough to the Shannon-Hartley limit that there is limited room for improvement, though 802.11ax saw fit to move to ODFMA. The third reduces the range (because there is less protection from multipath interference) but may increase the data rate by ~10%; 802.11ax introduced support for dynamic changing of guard-interval and symbol size. The largest speed increases are obtained the last two items in the list.

The channel width is increased by adding additional 5 MHz channels. For example, the 65 Mbps bit rate above for 802.11n is for a nominal frequency range of 20 MHz, comparable to that of 802.11g. However, in areas with minimal competition from other signals, 802.11n supports using a 40 MHz frequency band; the bit rate then goes up to 135 Mbps (or 150 Mbps if a smaller guard interval is used). This amounts to using two of the three available 2.4 GHz Wi-Fi bands. Similarly, the wide range in 802.11ac bit rates reflects support for using channel widths ranging from 20 MHz up to 160 MHz (32 5-MHz official channels).

Using multiple spatial streams is the newest data-rate-improvement technique; see *3.7.3 Multiple Spatial Streams*.

For all the categories in the table above, additional bits are used for error-correcting codes. For 802.11g operating at 54 Mbps, for example, the actual raw bit rate is $(4/3) \times 54 = 72$ Mbps, sent in symbols consisting of six bits as a unit.

3.7.1 Wi-Fi and Collisions

We looked extensively at the 10 Mbps Ethernet collision-handling mechanisms in *2.1 10-Mbps Classic Ethernet*, only to conclude that with switches and full-duplex links, Ethernet collisions are rapidly becoming a thing of the past. Wi-Fi, however, has brought collisions back from obscurity. An Ethernet sender will discover a collision, if one occurs, during the first slot time, by monitoring for faint interference with its own transmission. However, as mentioned in *3.6.2 Collisions*, Wi-Fi transmitting stations simply cannot detect collisions in progress. If another station transmits at the same time, a Wi-Fi sender will see nothing amiss although its signal will not be received. While there *is* a largely-collision-free mode for Wi-Fi operation (*3.7.7 Wi-Fi Polling Mode*), it is not commonly used, and collision management has a significant impact on ordinary Wi-Fi performance.

3.7.1.1 Link-Layer ACKs

The first problem with Wi-Fi collisions is even detecting them. Because of the inability to detect collisions directly, the Wi-Fi protocol adds **link-layer ACK packets**, at least for unicast transmission. These ACKs are our first example of Wi-Fi **control** packets and are unrelated to the higher-layer TCP ACKs.

The reliable delivery of these link-layer ACKs depends on careful timing. There are three time intervals applicable (numeric values here are for 802.11b/g in the 2.4 GHz band). The value we here call IFS is more formally known as DIFS (D for "distributed"; see *3.7.7 Wi-Fi Polling Mode*).

- slot time: 20 µsec
- IFS, the "normal" InterFrame Spacing: 50 µsec
- SIFS, the *short* IFS: 10 µsec

For comparison, note that the RTT between two Wi-Fi stations 100 meters apart is less than 1 µsec. At 11 Mbps, one IFS time is enough to send about 70 bytes; at 54 Mbps it is enough to send almost 340 bytes.

Once a station has received a data packet addressed to it, it waits for time SIFS and sends its ACK. At this point in time the receiver will be the only station authorized to send, because, as we will see in the next section, all other stations (including those on someone else's overlapping Wi-Fi) will be required to wait the longer IFS period following the end of the previous data transmission. These other stations will see the ACK before the IFS time has elapsed and will thus not interfere with it (though see exercise 4.0).

If a packet is involved in a collision, the receiver will send no ACK, so the sender will know something went awry. Unfortunately, the sender will *not* be able to tell whether the problem was due to a collision, or electromagnetic interference, or signal blockage, or excessive distance, or the receiver's being powered off. But as a collision is usually the most likely cause, and as assuming the lost packet was involved in a collision results in, at worst, a slight additional delay in retransmission, a collision will always be assumed.

Link-Layer ACKs contain no information – such as a sequence number – that identifies the packet being acknowledged. These ACKs simply acknowledge the most recent transmission, the one that ended one SIFS earlier. In the Wi-Fi context, this is unambiguous. It may be compared, however, to *6.1 Building Reliable Transport: Stop-and-Wait*, where at least one bit of packet sequence numbering is required.

3.7.1.2 Collision Avoidance and Backoff

The Ethernet collision-management algorithm was known as CSMA/CD, where CD stood for Collision Detection. The corresponding Wi-Fi mechanism is CSMA/**CA**, where CA stands for Collision **Avoidance**. A collision is presumed to have occurred if the link-layer ACK is not received. As with Ethernet, there is an exponential-backoff mechanism as well, though it is scaled somewhat differently.

Any sender wanting to send a new data packet waits the IFS time after first sensing the medium to see if it is idle. If no other traffic is seen in this interval, the station may then transmit immediately. However, if other traffic *is* sensed, the sender must do an exponential backoff even for its first transmission attempt; other stations, after all, are likely also waiting, and avoiding an initial collision is strongly preferred.

The initial backoff is to choose a random k<2^5 = 32 (recall that classic Ethernet in effect chooses an initial backoff of k<2^0 = 1; *ie* k=0). The prospective sender then waits k slot times. While waiting, the sender continues to monitor for other traffic; if any other transmission is detected, then the sender "suspends" the backoff-wait clock. The clock resumes when the other transmission has completed and one followup idle interval of length IFS has elapsed.

Note that, under these rules, data-packet senders *always* wait for at least one idle interval of length IFS before sending, thus ensuring that they never collide with an ACK sent after an idle interval of only SIFS.

On an Ethernet, if two stations are waiting for a third to finish before they transmit, they will both transmit as soon as the third is finished and so there will always be an initial collision. With Wi-Fi, because of the larger initial k<32 backoff range, such initial collisions are unlikely.

If a Wi-Fi sender believes there has been a collision, it retries its transmission, after doubling the backoff range to 64, then 128, 256, 512, 1024 and again 1024. If these seven attempts all fail, the packet is discarded and the sender starts over.

In one slot time, radio signals move 6,000 meters; the Wi-Fi slot time – unlike that for Ethernet – has nothing to do with the physical diameter of the network. As with Ethernet, though, the Wi-Fi slot time represents the fundamental unit for backoff intervals.

Finally, we note that, unlike Ethernet collisions, Wi-Fi collisions are a local phenomenon: if A and B transmit simultaneously, a collision occurs at node C only if the signals of A and B are both strong enough at C to interfere with one another. It is possible that a collision occurs at station C midway between A and B, but not at station D that is close to A. We return to this below in *3.7.1.4 Hidden-Node Problem*.

3.7.1.3 Wi-Fi RTS/CTS

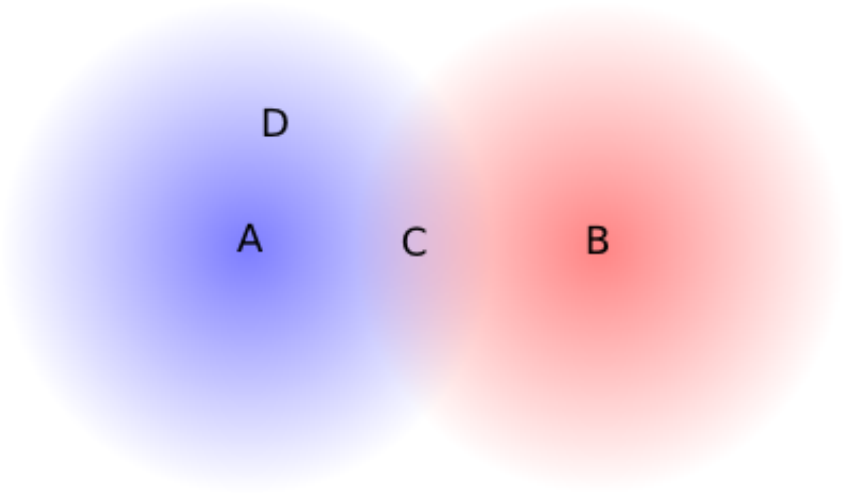
Wi-Fi stations optionally also use a request-to-send/clear-to-send (**RTS/CTS**) protocol, again negotiated with designated control packets. Usually this is used only for larger data packets; often, the RTS/CTS "threshold" (the size of the largest packet *not* sent using RTS/CTS) is set (as part of the Access Point configuration, *3.7.4 Access Points*) to be the maximum packet size, effectively disabling this feature. The

idea behind RTS/CTS is that a large packet that is involved in a collision represents a significant waste of potential throughput; for large packets, we should ask first.

The RTS control packet – which is small – is sent through the normal procedure outlined above; this packet includes the identity of the destination and the size of the data packet the station desires to transmit. The destination station then replies with CTS after the SIFS wait period, effectively preventing any other transmission after the RTS. The CTS packet also contains the data-packet size. The original sender then waits for SIFS after receiving the CTS, and sends the packet. If all other stations can hear both the RTS and CTS messages, then once the RTS and CTS are sent successfully no collisions should occur during packet transmission, again because the only idle times are of length SIFS and other stations should be waiting for time IFS.

3.7.1.4 Hidden-Node Problem

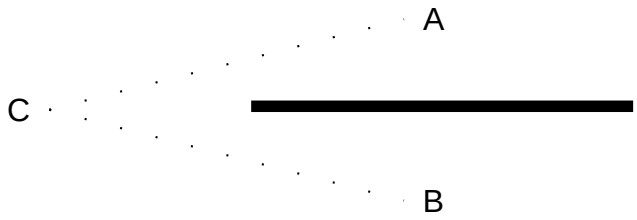
Consider the diagram below. Each station has a 100-meter range. Stations A and B are 150 meters apart and so cannot hear one another at all; each is 75 meters from C. If A is transmitting and B senses the medium in preparation for its own transmission, as part of collision avoidance, then B will conclude that the medium is idle and will go ahead and send.



However, C is within range of both A and B. If A and B transmit simultaneously, then from C's perspective a collision occurs. C receives nothing usable. We will call this a **hidden-node collision** as the senders A and B are hidden from one another; the general scenario is known as the **hidden-node problem**.

Note that node D receives only A's signal, and so no collision occurs at D.

The hidden-node problem can also occur if A and B cannot receive one another's transmissions due to a physical obstruction such as a radio-impermeable wall:



One of the rationales for the RTS/CTS protocol is the prevention of hidden-node collisions. Imagine that, instead of transmitting its data packet, A sends an RTS packet, and C responds with CTS. B has not heard the RTS packet from A, but *does* hear the CTS from C. A will begin transmitting after a SIFS interval, but B will not hear A's transmission. However, B will still wait, because the CTS packet contained the data-packet size and thus, implicitly, the length of time all other stations should remain idle. Because RTS packets are quite short, they are much less likely to be involved in collisions themselves than data packets.

3.7.1.5 Wi-Fi Fragmentation

Conceptually related to RTS/CTS is Wi-Fi **fragmentation**. If error rates or collision rates are high, a sender can send a large packet as multiple fragments, each receiving its own link-layer ACK. As we shall see in *5.3.1 Error Rates and Packet Size*, if bit-error rates are high then sending several smaller packets often leads to fewer total transmitted bytes than sending the same data as one large packet.

Wi-Fi packet fragments are reassembled by the receiving node, which may or may not be the final destination.

As with the RTS/CTS threshold, the fragmentation threshold is often set to the size of the maximum packet. Adjusting the values of these thresholds is seldom necessary, though might be appropriate if monitoring revealed high collision or error rates. Unfortunately, it is essentially impossible for an individual station to distinguish between reception errors caused by collisions and reception errors caused by other forms of noise, and so it is hard to use reception statistics to distinguish between a need for RTS/CTS and a need for fragmentation.

3.7.2 Dynamic Rate Scaling

Wi-Fi senders, if they detect transmission problems, are able to reduce their transmission bit rate in a process known as **rate scaling** or **rate control**. The idea is that lower bit rates will have fewer noise-related errors, and so as the error rate becomes unacceptably high – perhaps due to increased distance – the sender should fall back to a lower bit rate. For 802.11g, the standard rates are 54, 48, 36, 24, 18, 12, 9 and 6 Mbps. Senders attempt to find the transmission rate that maximizes throughput; for example, 36 Mbps with a packet loss rate of 25% has an effective throughput of $36 \times 75\% = 27$ Mbps, and so is better than 24 Mbps with no losses.

Senders may update their bit rate on a per-packet basis; senders may also choose different bit rates for different recipients. For example, if a sender sends a packet and receives no confirming link-layer ACK, the sender may fall back to the next lower bit rate. The actual bit-rate-selection algorithm lives in the particular Wi-Fi driver in use; different nodes in a network may use different algorithms.

The earliest rate-scaling algorithm was Automatic Rate Fallback, or ARF, *[KM97]*. The rate decreases after two consecutive transmission failures (that is, the link-layer ACK is not received), and increases after ten transmission successes.

A significant problem for rate scaling is that a packet loss may be due either to low-level random noise (white noise, or thermal noise) or to a collision (which is also a form of noise, but less random); only in the first case is a lower transmission rate likely to be helpful. If a larger number of collisions is experienced, the longer packet-transmission times caused by the lower bit rate may *increase* the frequency of hidden-node collisions. In fact, a *higher* transmission rate (leading to shorter transmission times) may help; enabling the RTS/CTS protocol may also help.

Signal Strength

Most Wi-Fi drivers report the received signal strength. Newer drivers use the IEEE Received Channel Power Indicator convention; the RCPI is an 8-bit integer proportional to the absolute power received by the antenna as measured in decibel-milliwatts (dBm). Wi-Fi values range from -10 dBm to -90 dBm and below. For comparison, the light from the star Polaris delivers about -97 dBm to one eye on a good night; Venus typically delivers about -73 dBm. A GPS satellite might deliver -127 dBm to your phone. (Inspired by Wikipedia on DBm.)

A variety of newer rate-scaling algorithms have been proposed; see *[JB05]* for a summary. One, Receiver-Based Auto Rate (RBAR, *[HVB01]*), attempts to incorporate the **signal-to-noise** ratio into the calculation of the transmission rate. This avoids the confusion introduced by collisions. Unfortunately, while the signal-to-noise ratio has a strong theoretical correlation with the transmission **bit-error rate**, most Wi-Fi radios will report to the host system the **received signal strength**. This is not the same as the signal-to-noise ratio, which is harder to measure. As a result, the RBAR approach has not been quite as effective in practice as might be hoped.

With the Collision-Aware Rate Adaptation algorithm (CARA, *[KKCQ06]*), a transmitting station attempts (among other things) to infer that its packet was lost to a collision rather than noise if, after one SIFS interval following the end of its packet transmission, no link-layer ACK has been received *and* the channel is still busy. This will detect collisions only when the colliding packet is *longer* than the station's own packet, and only when the hidden-node problem isn't an issue.

Because the actual data in a Wi-Fi packet may be sent at a rate not every participant is close enough to receive correctly, every Wi-Fi transmission begins with a brief preamble at the minimum bit rate. Link-layer ACKs, too, are sent at the minimum bit rate.

3.7.3 Multiple Spatial Streams

The latest innovation in improving Wi-Fi (and other wireless) data rates is to support multiple simultaneous data streams, through an antenna technique known as multiple-input-multiple-output, or **MIMO**. To use N streams, both sender and receiver must have N antennas; all the antennas use the same frequency channels but each transmitter antenna sends a different data stream. At first glance, any significant improvement in throughput might seem impossible, as the antenna elements in the respective sending and receiving groups are each within about half a wavelength of each other; indeed, in clear space MIMO is not possible.

The reason MIMO works in most everyday settings is that it puts multipath interference to positive use.

Consider again at the right-hand side of the final image of *3.6.6 Multipath*, in which the signal strength varies according to the blue ripples; the peaks and valleys have a period of about half a wavelength. We will assume initially that the signal strength is low enough that reception in the darkest blue areas is no longer viable; a single antenna with the misfortune to be in one of these "dead zones" may receive nothing.

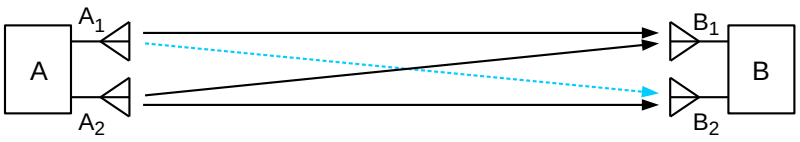
We will start with two simpler cases: SIMO (single-input-multiple-output) and MISO (multiple-input-single-output). In SIMO, the receiver has multiple antennas; in MISO, the transmitter. Assume for the moment that the multiple-antenna side has two antennas. In the simplest implementation of SIMO, the receiver picks the stronger of the two received signals and uses that alone; as long as at least one antenna is not in a "dead zone", reception is successful. With two antennas under half a wavelength apart, the odds are that at least one of them will be located outside a dead zone, and will receive an adequate signal.

Similarly, in simple MISO, the transmitter picks whichever of its antennas that gets a stronger signal to the receiver. The receiver is unlikely to be in a dead zone for *both* transmitter antennas. Note that for MISO the sender must get some feedback from the receiver to know which antenna to use.

We can do quite a bit better if signal-processing techniques are utilized so the two sender or two receiver antennas can be used simultaneously (though this complicates the mathematics considerably). Such signal-processing is standard in 802.11n and above; the Wi-Fi header, to assist this process, includes added management packets and fields for reporting MIMO-related information. One station may, for example, send the other a sequence of *training symbols* for discerning the response of the antenna system.

MISO with these added techniques is sometimes called **beamforming**: the sender coordinates its multiple antennas to maximize the signal reaching one particular receiver.

In our simplistic description of SIMO and MIMO above, in which only one of the multiple-antenna-side antennas is actually used, we have suggested that the idea is to improve marginal reception. At least one antenna on the multiple-antenna side can successfully communicate with the single antenna on the other side. **MIMO**, on the other hand, can be thought of as applying when transmission conditions are quite good all around, and every antenna on one side can reach every antenna on the other side. The key point is that, in an environment with a significant degree of multipath interference, the antenna-to-antenna paths may all be *independent*, or *uncorrelated*. At least one receiving antenna must be, from the perspective of at least one transmitting antenna, in a multipath-interference "gray zone" of reduced signal strength.



MIMO antennas

As a specific example, consider the diagram above, with two sending antennas A_1 and A_2 at the left and two receiving antennas B_1 and B_2 at the right. Antenna A_1 transmits signal S_1 and A_2 transmits S_2. There are thus four physical signal paths: A_1-to-B_1, A_1-to-B_2, A_2-to-B_1 and A_2-to-B_2. If we assume that the signal along the A_1-to-B_2 path (dashed and blue) arrives with half the strength of the other three paths (solid and black), then we have

signal received by B_1: $S_1 + S_2$
signal received by B_2: $S_1/2 + S_2$

From these, B can readily solve for the two independent signals S_1 and S_2. These signals are said to form two

spatial streams, though the spatial streams are abstract and do not correspond to any of the four physical signal paths.

The antennas are each more-or-less omnidirectional; the signal-strength variations come from multipath interference and not from physical aiming. Similarly, while the diagonal paths A_1-to-B_2 and A_2-to-B_1 are slightly longer than the horizontal paths A_1-to-B_1 and A_2-to-B_2, the difference is not nearly enough to allow B to solve for the two signals.

In practice, overall data-rate improvement over a single antenna can be considerably less than a factor of 2 (or than N, the number of antennas at each end).

The 802.11n standard allows for up to four spatial streams, for a theoretical maximum bit rate of 600 Mbps. 802.11ac allows for up to eight spatial streams, for an even-more-theoretical maximum of close to 7 Gbps. MIMO support is sometimes described with an $A \times B \times C$ notation, *eg* $3 \times 3 \times 2$, where A and B are the number of transmitting and receiving antennas and $C \leqslant \min(A,B)$ is the number of spatial streams.

3.7.4 Access Points

There are two standard Wi-Fi configurations: **infrastructure** and **ad hoc**. The former involves connection to a designated **access point**; the latter includes individual Wi-Fi-equipped nodes communicating informally. For example, two laptops can set up an ad hoc connection to transfer data at a meeting. Ad hoc connections are often used for very simple networks *not* providing Internet connectivity. Complex ad hoc networks are, however, certainly possible; see *3.7.8 MANETs*.

The **infrastructure** configuration is much more common. Stations in an infrastructure network communicate directly *only* with their access point, which, in turn, communicates with the outside world. If Wi-Fi nodes B and C share access point AP, and B wishes to send a packet to C, then B first forwards the packet to AP and AP then forwards it to C. While this introduces a degree of inefficiency, it does mean that the access point and its associated nodes automatically act as a true LAN: every node can reach every other node. (It is also often the case that most traffic is between Wi-Fi nodes and the outside world.) In an ad hoc network, by comparison, it is common for two nodes to be able to reach each other only by forwarding through an intermediate third node; this is in fact a form of the hidden-node scenario.

Wi-Fi access points are generally identified by their **SSID** ("Service Set IDentifier"), an administratively defined human-readable string such as "linksys" or "loyola". Ad hoc networks also have SSIDs; these are generated pseudorandomly at startup and look like (but are not) 48-bit MAC addresses.

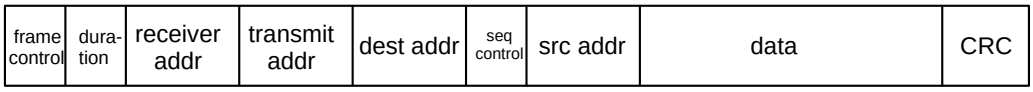
Portable Access Points

Being a Wi-Fi access point is a very specific job; Wi-Fi-enabled "station" devices like phones and workstations do not generally act as access points. However, it is often possible to for a station device to become an access point if the access-point mode is supported by the underlying radio hardware, and if suitable drivers can be found. The Linux `hostapd` package is one option. The FCC may or may not bestow its blessing.

Many access points can support multiple SSIDs simultaneously. For example, an access point might support SSID "guest" with limited authentication (below), and also SSID "secure" with much stronger authentication.

Finally, Wi-Fi is by design completely interoperable with Ethernet; if station A is associated with access point AP, and AP also connects via (cabled) Ethernet to station B, then if A wants to send a packet to B it sends it using AP as the Wi-Fi destination but with B also included in the header as the "actual" destination. Once it receives the packet by wireless, AP acts as an Ethernet switch and forwards the packet to B. While this forwarding is transparent to senders, the Ethernet and Wi-Fi LAN header formats are quite different.

Ethernet



Wi-Fi Data

The above diagram illustrates an Ethernet header and the Wi-Fi header for a typical data packet (not using Wi-Fi quality-of-service features). The Ethernet type field usually moves to an IEEE Logical Link Control header in the Wi-Fi region labeled "data". The receiver and transmitter addresses are the MAC addresses of the nodes receiving and transmitting the (unicast) packet; these may each be different from the ultimate destination and source addresses. If station B wants to send a packet to station C in the same network, the source and destination are B and C but the transmitter and receiver are B and the access point. In infrastructure mode either the receiver or transmitter address is always the access point; in typical situations either the receiver is the destination or the sender is the transmitter. In ad hoc mode, if LAN-layer routing is used then all four addresses may be distinct; see *3.7.8.1 Routing in MANETs*.

3.7.4.1 Joining a Network

To join the network, an individual station must first discover its access point, and must **associate** and then **authenticate** to that access point before general communication can begin. (Older forms of authentication – so-called "open" authentication and the now-deprecated WEP authentication – came before association, but newer authentication protocols such as WPA2, WPA2-Personal and WPA2-Enterprise (*3.7.5 Wi-Fi Security*) come after.) We can summarize the stages in the process as follows:

- scanning (or active probing)
- open-authentication and association
- true authentication
- DHCP (getting an IP address, *7.10 Dynamic Host Configuration Protocol (DHCP)*)

The association and authentication processes are carried out by an exchange of special **management packets**, which are confined to the Wi-Fi LAN layer. Occasionally stations may re-associate to their Access Point, *eg* if they wish to communicate some status update.

Access points periodically broadcast their SSID in special **beacon** packets (though for pseudo-security reasons the SSID in the beacon packets can be suppressed). Beacon packets are one of several Wi-Fi-layer-only **management packets**; the default beacon-broadcast interval is 100 ms. These broadcasts allow stations to

see a list of available networks; the beacon packets also contain other Wi-Fi network parameters such as radio-modulation parameters and available data rates.

Another use of beacons is to support the power-management **doze** mode. Some stations may elect to enter this power-conservation mode, in which case they inform the access point, record the announced beacon-transmission time interval and then wake up briefly to receive each beacon. Beacons, in turn, each contain a list (in a compact bitmap form) of each dozing station for which the access point has a packet to deliver.

Ad hoc networks have beacon packets as well; all nodes participate in the regular transmission of these via a distributed algorithm.

A connecting station may either wait for the next scheduled beacon, or send a special **probe-request** packet to elicit a beacon-like probe-response packet. These operations may involve listening to or transmitting on multiple radio channels, sequentially, as the station does not yet know the correct channel to use. Unconnected stations often send probe-request packets at regular intervals, to keep abreast of available networks; it is these probe packets that allow tracking by the station's MAC address. See *3.7.4.2 MAC Address Randomization*.

Once the beacon is received, the station initiates an **association** process. There is still a vestigial open-authentication process that comes before association, but once upon a time this could also be "shared WEP key" authentication (below). Later, support for a wide range of authentication protocols was introduced, via the 802.1X framework; we return to this in *3.7.5 Wi-Fi Security*. For our purposes here, we will include open authentication as part of the association process.

Wi-Fi Drivers

Even in 2015, 100%-open-source Wi-Fi drivers are available only for selected hardware, and even then not all operations may be supported. Something as simple in principle as changing one's source Wi-Fi MAC address is sometimes not possible, though see *3.7.4.2 MAC Address Randomization*. Using multiple MAC addresses for a host plus embedded virtual machines is another problematic case.

In open authentication the station sends an authentication request to the access point and the access point replies. About all the station needs to know is the SSID of the access point, though it is usually possible to configure the access point to restrict admission to stations with MAC (physical) addresses on a predetermined list. Stations sometimes evade MAC-address checking by changing their MAC address to an acceptable one, though some Wi-Fi drivers do not support this.

Because the SSID plays something of the role of a password here, some Wi-Fi access points are configured so that beacon packets does not contain the SSID; such access points are said to be **hidden**. Unfortunately, access points hidden this way are easily unmasked: first, the SSID is sent in the clear by any other stations that need to authenticate, and second, an attacker can often transmit forged deauthentication or disassociation requests to force legitimate stations to retransmit the SSID. (See "management frame protection" in *3.7.5 Wi-Fi Security* for a fix to this last problem.)

The shared-WEP-key authentication was based on the (obsolete) WEP encryption mechanism (*3.7.5 Wi-Fi Security*). It involved a challenge-response exchange by which the station proved to the access point that it knew the shared WEP key. Actual WEP encryption would then start slightly later.

Once the open-authentication step is done, the next step in an infrastructure network is for the station to **associate** to the access point. This involves an association request from station to access point, and an

association response in return. The primary goal of the association exchange is to ensure that the access point knows (by MAC address) what stations it can reach. This tells the access point how to deliver packets to the associating station that come from other stations or the outside world. Association is not necessary in an ad hoc network.

The entire connection process (including secure authentication, below, and DHCP, *7.10 Dynamic Host Configuration Protocol (DHCP)*), often takes rather longer than expected, sometimes several seconds. See *[PWZMTQ17]* for a discussion of some of the causes. Some station and access-point pairs appear not to work as well together as other pairs.

3.7.4.2 MAC Address Randomization

Most Wi-Fi-enabled devices are configured to transmit Wi-Fi **probe requests** at regular intervals (and on all available channels), at least when not connected. These probe requests identify available Wi-Fi networks, but they also reveal the device's MAC address. This allows sites such as stores to track customers by their device. To prevent such tracking, some devices now support **MAC address randomization**, proposed in *[GG03]*: the use at appropriate intervals of a new MAC address randomly selected by the device.

Probe requests are generally sent when the device is not joined to a network. To prevent tracking via probe requests, the simplest approach is to change the MAC address used for probes at regular, frequent intervals. A device might even change its MAC address on every probe.

Changing the MAC address used for actually *joining* a network is also important to prevent tracking, but introduces some complications. **RFC 7844** suggests these options for selecting new random addresses:

- At regular **time intervals**
- **Per connection**: each time the device connects to a Wi-Fi network, it will select a new MAC address
- **Per network**: like the above, except that if the device reconnects to the same network (identified by SSID), it will use the same MAC address

The first option, changing the joined MAC address at regular time intervals, breaks things. First, it will likely result in assignment of a new IP address to the device, terminating all existing connections. Second, many sites still authenticate – at least in part – based on the MAC address. The per-connection option prevents the first problem. The per-network option prevents both, but allows a site at which the device actually joins the network to track repeat connections. (Configuring the device to "forget" the connection between successive joins will usually prevent this, but may not be convenient.)

Another approach to the tracking problem is to disable probe requests entirely, except on explicit demand.

Wi-Fi MAC address randomization is, unfortunately, not a complete barrier to device tracking; there are other channels through which devices may leak information. For example, probe requests also contain device-capability data known as Information Elements; these values are often distinctive enough that they allow at least partial fingerprinting. Additionally, it is possible to track many Wi-Fi devices based on minute variations in the modulated signals they transmit. MAC address randomization does nothing to prevent such "radiometric identification". Access points can also impersonate other popular access points, and thus trick devices into initiating a connection with their real MAC addresses. See *[BBGO08]* and *[VMCCP16]* for these and other examples.

Finally, MAC address randomization may have applications for Ethernet as well as Wi-Fi. For example, in the original IPv6 specification, IPv6 addresses embedded the MAC address, and thus introduced the

possibility of tracking a device by its IPv6 address. MAC address randomization can prevent this form of tracking as well. However, other techniques implementable solely in the IPv6 layer appear to be more popular; see *8.2.1 Interface identifiers*.

3.7.4.3 Roaming

Large installations with multiple access points can create "roaming" access by assigning all the access points the same SSID. An individual station will stay with the access point with which it originally associated until the signal strength falls below a certain level (as determined by the station), at which point it will seek out other access points with the same SSID and with a stronger signal. In this way, a large area can be carpeted with multiple Wi-Fi access points, so as to look like one large Wi-Fi domain. The access points are often connected via a wired LAN, known as the **distribution system**, though the use of Wi-Fi itself to provide interconnection between access points is also an option (*3.7.4.4 Mesh Networks*). At any one time, a station may be associated to only one access point. In 802.11 terminology, a multiple-access-point configuration with a single SSID is known as an "extended service set" or **ESS**.

In order for such a large-area network to work, traffic *to* a wireless station, *eg* B, must find that station's current access point, *eg* AP. To help the distribution system track B's current location, B is required, at the time it moves from AP_{old} to AP, to send to AP a **reassociation request**, containing AP_{old}'s address. This sets in motion a number of things; one of them is that AP contacts AP_{old} to verify (and terminate) the former association. This reassociation process also gives AP an opportunity – not spelled out in detail in the standard – to notify the distribution system of B's new location.

If the distribution system is a switched Ethernet supporting the usual learning mechanism (*2.4 Ethernet Switches*), one simple approach to roaming stations is to handle this the same way as, in a wired Ethernet, traffic finds a laptop that has been unplugged, carried to a new location, and plugged in again. Suppose our wireless node B has been exchanging packets via the distribution system with wired node C (perhaps a router connecting B to the Internet). When B moves from AP_{old} to AP, all it has to do is send any packet over the LAN to C, and the Ethernet switches on the path from B to C will then learn the route through the switched Ethernet from C back to B's new AP, and thus to B. It is also possible for B's new AP to send this switch-updating packet, perhaps as part of its reassociation response.

This process may leave other switches in the distribution system still holding in their forwarding tables the old location for B. This is not terribly serious, as it will be fixed for any one switch as soon as B sends a packet to a destination reached by that switch. The problem can be avoided entirely if, after moving, B (or, again, its new AP) sends out an Ethernet broadcast packet.

Running Ethernet cable to remote access points can easily dwarf the cost of the access point itself. As a result, there is considerable pressure to find ways to allow the Wi-Fi network itself to form the distribution system. We return to this below, in *3.7.4.4 Mesh Networks*.

The IEEE 802.11r amendment introduced the standardization of **fast handoffs** from one access point to another within the same ESS. It allows the station and new access point to reuse the same pairwise master keys (below) that had been negotiated between the station and the old access point. It also slightly streamlines the reassociation process. Transitions must, however, still be initiated by the station. The amendment is limited to handoffs; it does not address finding the access point to which a particular station is associated, or routing between access points.

Because handoffs must be initiated by the station, sometimes all does not quite work out smoothly. Within an ESS, most newer devices (2018) are quite good at initiating handoffs. However, this is not always the

case for older devices, and is usually still not the case for many mobile-station devices moving from one ESS to another (that is, where there is a change in the SSID). Such devices may cling to their original access point well past the distance at which the original signal ceases to provide reasonable bandwidth, as long as it does not vanish entirely. Many Wi-Fi "repeaters" or "extenders" (below) sold for residential use do require a second SSID, and so will often do a poor job at supporting roaming.

Some access points support proprietary methods for dealing with older mobile stations that are reluctant to transfer to a closer access point within the same ESS, though these techniques are now seldom necessary. By communicating amongst themselves, the access points can detect when a station's signal is weak enough that a handoff would be appropriate. One approach involves having the original access point initiate a dissociation. At that point the station will reconnect to the ESS but should now connect to an access point within the ESS that has a stronger signal. Another approach involves having the access points all use the same MAC address, so they are indistinguishable. Whichever access point receives the strongest signal from the station is the one used to transmit *to* the station.

3.7.4.4 Mesh Networks

Being able to move freely around a multiple-access-point Wi-Fi installation is very important for many users. When such a Wi-Fi installation is created in a typical office building pre-wired for Ethernet, the access points all plug into the Ethernet, which becomes the distribution network. However, in larger-scale residential settings, and even many offices, running Ethernet cable may be prohibitively expensive. In such cases it makes sense to have the access points interconnect via Wi-Fi itself. If Alice associates to access point A and sends a packet destined for the outside world, then perhaps A will have to forward the packet to Wi-Fi node B, which will in turn forward it to C, before delivery can be complete.

This is sometimes easier said than done, however, as the original Wi-Fi standards did not provide for the use of Wi-Fi access points as "repeaters"; there was no standard mechanism for a Wi-Fi-based distribution network.

One inexpensive approach is to use devices sometimes sold as Wi-Fi "extenders". Such devices typically set up a new SSID, and forward all traffic to the original SSID. Multi-hop configurations are possible, but must usually be configured manually. Because the original access point and the extender have different SSIDs, many devices will not automatically connect to whichever is closer, preferring to stick to the SSID originally connected to until that signal essentially disappears completely. This is, for many mobile users, reason enough to give up on this strategy.

The desire for a Wi-Fi-based distribution network has led to multiple proprietary solutions. It is possible to purchase a set of Wi-Fi "mesh routers" (2018), often sold at a considerable premium over "standard" routers. After they are set up, these generally work quite well: they present a single SSID, and support fast handoffs from one access point to another, without user intervention. To the user, coverage appears seamless.

The downside of a proprietary mechanism, however, is that once you buy into one solution, equipment from other vendors will seldom interoperate. This has led to pressure for standardization. The IEEE introduced "mesh networking" in its 802.11s amendment, finalized as part of the 2012 edition of the full 802.11 standard; it was slow to catch on. The Wi-Fi Alliance introduced the Wi-Fi EasyMesh solution in 2018, based on 802.11s, but, as of the initial rollout, no vendors were yet on board.

We will assume, for the time being, that Wi-Fi mesh networking is restricted to the creation of a distribution network interconnecting the access points; ordinary stations do not participate in forwarding other users'

packets through the mesh. Common implementations often take this approach, but in fact the 802.11s amendment allows more general approaches.

In creating a mesh network with a Wi-Fi distribution system – proprietary or 802.11s – the participating access points must address the following issues:

- They must authenticate to one another
- They must identify the correct access point to reach a given station B
- They must correctly handle station B's movement to a different access point
- They must agree on how to route, through the mesh of access points, between the station and the connection to the Internet

Eventually the routing issue becomes the same routing problem faced by MANETs (*3.7.8 MANETs*), although restricted to the (simpler) case where all nodes are under common management. Routing is not trivial; the path A→B→C might be shorter than the alternative path A→D→E→C, but support a lower data rate.

The typical 802.11s solution is to have the multiple access points participate in a **mesh BSS**. This allows all the access points to appear to be on a single LAN. In this setting, the mesh BSS is separate from the ESS seen by the user stations, and is only used for inter-access-point communication.

One (or more) access points are typically connected to the Internet; these are referred to as **root mesh stations**.

In the 802.11s setting, mesh discovery is achieved via initial configuration of a mesh SSID, together with a WPA3 passphrase. Mutual authentication is then via WPA3, below; it is particularly important that each pair of stations authenticate symmetrically to one another.

If station B associates to access point AP, then AP uses the mesh BSS to deliver packets sent by B to the root mesh station (or to some other AP). For reverse traffic, B's reassociation request sent to AP gives AP an opportunity to interact with the mesh BSS to update B's new location. The act of B's sending a packet via AP will also tell the mesh BSS how to find B.

Routing through the mesh BSS is handled via the HWMP protocol, *9.4.3 HWMP*. This protocol typically generates a tree of station-to-station links (that is, a subset of all links that contains no loops), based at the root station. This process uses a routing metric that is tuned to the wireless environment, so that high-bandwidth and low-error links are preferred.

If a packet is routed through the mesh BSS from station A to station B, then more addresses are needed in the packet header. The ultimate source and destination are A and B, and the transmitter and receiver correspond to the specific hop, but the packet also needs a source and destination within the mesh, perhaps corresponding to the two access points to which A and B connect. 802.11s handles this by adding a **mesh control field** consisting of some management fields (such as TTL and sequence number) and a variable-length block of up to three additional addresses.

It is also possible for ordinary stations to join the 802.11s mesh BSS directly, rather than restricting the mesh BSS to the access points. This means that the stations will participate in the mesh as routing nodes. It is hard to predict, in 2018, how popular this will become.

The EasyMesh standard of the Wi-Fi Alliance is not exactly the same as the IEEE 802.11s standard. For one thing, the EasyMesh standard specifies that one access point – the one connected to the Internet – will be a

"Multi-AP" Controller; the other access points are called Agents. The EasyMesh standard also incorporates parts of the IEEE 1905.1 standard for home networks, which simplifies initial configuration.

3.7.5 Wi-Fi Security

Unencrypted Wi-Fi traffic is visible to anyone nearby with an appropriate receiver; this eavesdropping zone can be expanded by use of a larger antenna. Because of this, Wi-Fi security is important, and Wi-Fi supports several types of traffic encryption.

The original – and now obsolete – Wi-Fi encryption standard was Wired-Equivalent Privacy, or **WEP**. It involved a 5-byte key, later sometimes extended to 13 bytes. The encryption algorithm was based on RC4, *22.7.4.1 RC4*. The key was a **pre-shared key**, manually configured into each station.

Because of the specific way WEP made use of the RC4 cipher, it contained a fatal (and now-classic) flaw. Bytes of the key could be could be "broken" – that is, guessed – sequentially. Knowing bytes 0 through i–1 would allow an attacker to guess byte i with a relatively small amount of data, and so on through the entire key. See *22.7.7 Wi-Fi WEP Encryption Failure* for details.

WEP was replaced with Wi-Fi Protected Access, or **WPA**. This used the so-called TKIP encryption algorithm that, like WEP, was ultimately based on RC4, but which was immune to the sequential attack that made WEP so vulnerable. WPA was later replaced by WPA2 as part of the IEEE 802.11i amendment, which uses the presumptively stronger AES encryption (*22.7.2 Block Ciphers*); the variant used by WPA2 is known as CCMP. WPA2 encryption is believed to be quite secure, although there was a vulnerability in the associated Wi-Fi Protected Setup protocol. In the 802.11i standard, WPA2 is known as the **robust security network** protocol. Access points supporting WPA or WPA2 declare this in their beacon and probe-response packets; these packets also include a list of acceptable ciphers.

WPA2 (and WPA) comes in two flavors: **WPA2-Personal** and **WPA2-Enterprise**. These use the same AES encryption, but differ in how keys are managed. WPA2-Personal, appropriate for many smaller sites, uses a pre-shared master key, known as the PSK. This key must be entered into the Access Point (ideally not over the air) and into each connecting station. The key is usually a secure hash (*22.6 Secure Hashes*) of a passphrase. The use of a single key for multiple stations makes changing the key, or revoking the key for a particular user, difficult.

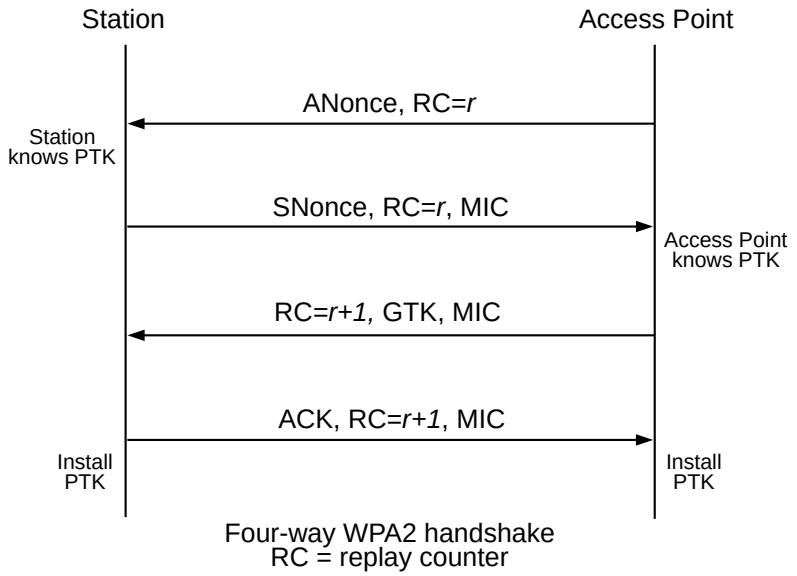
In 2018, the IEEE introduced **WPA3**, intended to fix a host of accumulated issues. Perhaps the most important change is that WPA3-Personal switches from the WPA2 four-way handshake to the SAE mutual-password-authentication mechanism, *22.8.2 Simultaneous Authentication of Equals*. We return to WPA3 below, at *3.7.5.3 WPA3*.

3.7.5.1 WPA2 Four-way handshake

In any secure Wi-Fi authentication protocol, the station must authenticate to the access point *and* the access point must authenticate to the station; without the latter part, stations might inadvertently connect to rogue access points, which would then likely gain at least partial access to private data. This bidirectional authentication is achieved through the so-called **four-way handshake**, which also generates a **session key**, known as the pairwise *transient* key or **PTK**, that is independent of the master key. Compromise of the PTK should not allow an attacker to determine the master key. To further improve security, the PTK is used to generate the temporal key, **TK**, used to encrypt data messages, a separate message-signing key used in the MIC code, below, and some management-packet keys.

In WPA2-Personal, the master key is the pre-shared key (PSK); in WPA2-Enterprise, below, the master key is the negotiated "pairwise master key", or PMK. The four-way handshake begins immediately after association and, for WPA2-Enterprise, the selection of the PMK. None of the four packets that are part of the handshake are encrypted.

Both station and access point begin by each selecting a random string, called a **nonce**, typically 32 bytes long. In the diagram below, the access point (authenticator) has chosen `ANonce` and the station (supplicant) has chosen `SNonce`. The PTK will be a secure hash of the master key, both nonces, and both MAC addresses. The first packet of the four-way handshake is sent by the access point to the station, and contains its nonce, unencrypted. This packet also contains a *replay counter*, RC; the access point assigns these sequentially and the station echoes them back.



At this point the station has enough information to compute the PTK; in the second message of the handshake it now sends its own nonce to the access point. The nonce is again sent in the clear, but this second message also includes a digital signature. This signature is sometimes called a Message Integrity Code, or MIC, and in the 802.11i standard is officially named *Michael*. It is calculated in a manner similar to the HMAC mechanism of *22.6.1 Secure Hashes and Authentication*, and uses its own key derived from the PTK.

Upon receipt of the station's nonce, the access point too is able to compute the PTK. With the PTK now in hand, the access point verifies the attached signature. If it checks out, that proves to the access point that the station did in fact know the master key, as a valid signature could not have been constructed without it. The station has now authenticated itself to the access point.

For the third stage of the handshake, the access point, now also in possession of the PTK, sends a signed message to the station. The replay counter is incremented, and an optional *group temporal key*, GTK, may be included for encrypting non-unicast messages. If the GTK is included, it is encrypted with the PTK, though the entire message is not encrypted. When this third message is received and verified, the access point has authenticated itself to the station. The fourth and final step is simply an acknowledgment from the client.

Four-way-handshake packets are sent in the EAPOL format, described in the following section. This format can be used to identify the handshake packets in WireShark scans.

One significant vulnerability of the four-way handshake when WPA2-Personal is used is that if an eavesdropper records the messages, then it can attempt an offline brute-force attack on the key. Different values of the passphrase used to generate the PSK can be tried until the MIC values computed for the second and third packets match the values in the corresponding recorded packets. At this point the attacker can not only authenticate to the network, but can also decrypt packets. This attack is harder with WPA2-Enterprise, as each user has a different key.

Other WPA2-Personal stations on the same network can also eavesdrop, given that all stations share the same PSK, and that the PTK is generated from the PSK and information transmitted without encryption. The Diffie-Hellman-Merkle key-exchange mechanism, *22.8 Diffie-Hellman-Merkle Exchange*, would avoid this difficulty; keys produced this way are *not* easily determined by an eavesdropper, even one with inside information about master keys. However, this was not used, in part because WPA needed to be rushed into service after the failure of WEP.

3.7.5.1.1 KRACK Attack

The purpose of the replay counter, RC in the diagram above, is to prevent an attacker from reusing an old handshake packet. Despite this effort, replayed or regenerated instances of the third handshake packet can sometimes be used to seriously weaken the underlying encryption. The attack, known as the **Key Reinstallation Attack**, or KRACK, is documented in *[VP17]*. The attack has several variations, some of which address a particular implementation's interpretation of the IEEE standard, and some of which address other Wi-Fi keys (*eg* the group temporal key) and key handshakes (*eg* the handshake used by *3.7.4.3 Roaming*). We consider only the most straightforward form here.

The ciphers used by WPA2 are all "stream" ciphers (*22.7.4 Stream Ciphers*), meaning that, for each packet, the key is used to generate a *keystream* of pseudorandom bits, the same length as the packet; the packet is then XORed with this keystream to encrypt it. It is essential for this scheme's security that the keystreams of different packets are unrelated; to achieve this, the keystream algorithm incorporates an *encryption nonce*, initially 1 and incremented for each successive packet.

The core observation of KRACK is that, whenever the station installs or reinstalls the PTK, it also resets this encryption nonce to 1. This has the effect of resetting the keystream, so that, for a while, each new packet will be encrypted with exactly the same keystream as an earlier packet.

This key reinstallation at the station side occurs whenever an instance of the third handshake packet arrives. Because of the possibility of lost packets, the handshake protocol must allow duplicates of any packet.

The basic strategy of KRACK is now to force key reinstallation, by arranging for either the access point or the attacker to deliver duplicates of the third handshake packet to the station.

In order to interfere with packet delivery, the attacker must be able to block and delay packets from the access point to the station, and be able to send its own packets to the station. The easiest way to accomplish this is for the attacker to be set up as a "clone" of the real access point, with the same MAC address, *but operating on a different Wi-Fi channel*. The attacker receives messages from the real access point on the original channel, and is able to selectively retransmit them to the station on the new channel. This can be described as a channel-based man-in-the-middle attack; *cf 22.9.3 Trust and the Man in the Middle*. Alternatively, the attacker may also be able to selectively jam messages from the access point.

If the attacker can block the fourth handshake packet, from station to access point, then the access point will eventually time out and retransmit a duplicate third packet, complete with properly updated replay counter.

The attacker can delay the duplicate third packet, if desired, in order to prolong the interval of keystream reuse. The station's response to this duplicate third packet will be encrypted, but the attacker can usually generate a forged, unencrypted version.

Forcing reuse of the keystream does not automatically break the encryption. However, in many cases the plaintext of a few packets can be guessed by context, and hence, by XORing, the keystream used to encrypt the packet can be determined. This allows trivial decryption of any later packet encrypted with the same keystream.

Other possibilities depend on the cipher. When the TKIP cipher is used, a vulnerability in the MIC algorithm may allow determination of the key used in the MIC; this in turn would allow the attacker to inject new packets, properly signed, into the connection. These new packets can be encrypted with one of the broken keystreams. This strategy does not work with AES (CCMP) encryption.

The KRACK vulnerability was fixed in `wpa_supplicant` by disallowing reinstallation of the same key. That is, if a retransmission of the third handshake packet is received, it is ignored; the encryption nonce is not reset.

3.7.5.2 WPA2-Enterprise

The **WPA2-Enterprise** alternative allows each station to have its own separate key. In fact, it largely separates the encryption mechanisms from the Wi-Fi protocols, allowing sites great freedom in choosing the former. Despite the "enterprise" in the name, it is also well suited for smaller sites. WPA2-Enterprise is based rather closely on the 802.1X framework, which supports arbitrary authentication protocols as plug-in modules.

In principle, the only improvement WPA2-Enterprise offers over WPA2-Personal is the ability to assign individual Wi-Fi passwords. In practice, this is an enormously important feature. It prevents, for example, one user from easily decrypting packets sent by another user.

The keys are all held by a single common system known as the **authentication server**, usually unrelated to the access point. The client node (that is, the Wi-Fi station) is known as the **supplicant**, and the access point is known as the **authenticator**.

To begin the authentication process, the supplicant contacts the authenticator using the Extensible Authentication Protocol, or **EAP**, with what amounts to a request to authenticate to that access point. EAP is a generic message framework meant to support multiple specific types of authentication; see **RFC 3748** and **RFC 5247**. The EAP request is forwarded to the authentication server, which may exchange (via the authenticator) several challenge/response messages with the supplicant. No secret credentials should be sent in the clear.

EAP is usually used in conjunction with the RADIUS (Remote Authentication Dial-In User Service) protocol (**RFC 2865**), which is a specific (but flexible) authentication-server protocol. WPA2-Enterprise is sometimes known as 802.1X mode, EAP mode or RADIUS mode (though WPA2-Personal is also based on 802.1X, and uses EAP in its four-way handshake).

EAP communication takes place before the supplicant is given an IP address; thus, a mechanism must be provided to support exchange of EAP packets between supplicant and authenticator. This mechanism is known as EAPOL, for EAP Over LAN. EAP messages between the authenticator and the authentication server, on the other hand, can travel via IP; in fact, sites may choose to have the authentication server

hosted remotely. Specific protocols using the EAP/RADIUS framework often use packet formats other than EAPOL, but EAPOL will be used in the concluding four-way handshake.

Once the authentication server (*eg* RADIUS server) is set up, specific per-user authentication methods can be entered. This can amount to ⟨username,password⟩ pairs (below), or some form of security certificate, or sometimes both. The authentication server will generally allow different encryption protocols to be used for different supplicants, thus allowing for the possibility that there is not a common protocol supported by all stations.

In WPA2-Enterprise, the access point no longer needs to know anything about what authentication protocol is actually used; it is simply the middleman forwarding EAP packets between the supplicant and the authentication server. In particular, the access point does not need to support any specific authentication protocol. The access point allows the supplicant to connect to the network once it receives permission to do so from the authentication server.

At the end of the authentication process, the supplicant and the authentication server will, as part of that process, also have established a shared secret. In WPA2-Enterprise terminology this is known as the **pairwise master key** or PMK. The authentication server then communicates the PMK securely to the access point (using any standard protocol; see *22.10 SSH and TLS*). The next step is for the supplicant and the access point to negotiate their session key. This is done using the four-way-handshake mechanism of the previous section, with the PMK as the master key. The resultant PTK is, as with WPA2-Personal, used as the session key.

WPA2-Enterprise authentication typically does require that the access point have an IP address, in order to be able to contact the authentication server. An access point using WPA2-Personal authentication does not need an IP address, though it may have one simply to enable configuration.

3.7.5.2.1 Enabling WPA2-Enterprise

Configuring a Wi-Fi network to use WPA2-Enterprise authentication is relatively straightforward, as long as an authentication server running RADIUS is available. We here give an outline of setting up WPA2-Enterprise authentication using FreeRADIUS (version 2.1.12, 2018). We want to enable per-user passwords, but *not* per-user certificates. Passwords will be stored on the server using SHA-1 hashing (*22.6 Secure Hashes*). This is not necessarily strong enough for production use; see *22.6.2 Password Hashes* for other options. Because passwords will be hashed, the client will have to communicate the actual password to the authentication server; authentication methods such as those in *22.6.3 CHAP* are not an option.

The first step is to set up the access point. This is generally quite straightforward; WPA2-Enterprise is supported even on inexpensive access points. After selecting the option to enable WPA2-Enterprise security, we will need to enter the IP address of the authentication server, and also a "shared secret" password for authenticating messages between the access point and the server (see *22.6.1 Secure Hashes and Authentication* for message-authentication techniques).

Configuration of the RADIUS server is a bit more complex, as both RADIUS and EAP are both quite general; both were developed long before 802.1X, and both are used in many other settings as well. Because we have decided to use hashed passwords – which implies the client station will send the plaintext password to the authentication server – we will need to use an authentication method that creates an encrypted tunnel. The Protected EAP method is well-suited here; it encrypts its traffic using TLS (*22.10.2 TLS*, though here without TCP). (There is also an EAP TLS method, using TLS directly and traditionally requiring client-side certificates, and a TTLS method, for Tunneled TLS.)

Within the PEAP encrypted tunnel, we want to use plaintext password authentication. Here we want the Password Authentication Protocol, PAP, which basically just asks for the username and password. FreeRADIUS does not allow PAP to run directly within PEAP, so we interpose the Generic Token Card protocol, GTC. (There is no "token card" device anywhere in sight, but GTC is indeed quite generic.)

We probably also have to tell the RADIUS server the IP address of the access point. The access point here must *have* an IP address, specifically for this communication.

We enable all these things by editing the `eap.conf` file so as to contain the following entries:

```
default_eap_type = peap

...

peap {
        default_eap_type = gtc
        ...
}

...

gtc {
        auth_type = PAP
        ...
}
```

The next step is to create a (username, hashed_password) credential pair on the server. To keep things simple, we will store credentials in the `users` file. The username will be "alice", with password "snorri". Per the FreeRADIUS rules, we need to convert the password to its SHA-1 hash, encoded using base64. There are several ways to do this; we will here make use of the OpenSSL command library:

```
echo -n "snorri" | openssl dgst -binary -sha1 | openssl base64
```

This returns the string `7E6FbhrN2TYOkrBti+8W8weC2W8=` which we then enter into the `users` file as follows:

```
alice   SHA1-Password := "7E6FbhrN2TYOkrBti+8W8weC2W8="
```

Other options include `Cleartext-Password`, `MD5-Password` and `SSHA1-Password`, with the latter being for salted passwords (which are recommended).

With this approach, Alice will have difficulty changing her password, unless she is administrator of the authentication server. This is not necessarily worse than WPA2-Personal, where Alice shares her password with other users. However, if we want to support user-controlled password changing, we can configure the RADIUS server to look for the (username, hashed_password) credentials in a database instead of the `users` file. It is then relatively straightforward to create a web-based interface for allowing users to change their passwords.

Now, finally, we try to connect. Any 802.1X client should ask for the username and password, before communication with the authentication server begins. Some may also ask for a preferred authentication method (though our RADIUS server here is only offering one), an optional certificate (which we are not using), or an "anonymous identity", which is a way for a client to specify a particular authentication server if there are

several. If all goes well, connection should be immediate. If not, FreeRADIUS has an authentication-testing tool, and copious debugging output.

3.7.5.3 WPA3

In 2018 the Wi-Fi Alliance introduced WPA3, a replacement for WPA2. The biggest change is that, when both parties are WPA3-aware, the WPA2 four-way handshake is replaced with SAE, *22.8.2 Simultaneous Authentication of Equals*. The advantage of SAE here is that an eavesdropper can get nowhere with an offline, dictionary-based, brute-force password attack; recall from the end of *3.7.5.1 WPA2 Four-way handshake* that WPA2 is quite vulnerable in this regard. An attacker can still attempt an *on*line brute-force attack on WPA3, *eg* by parking a van within Wi-Fi range and trying one password after another, but this is slow.

Another consequence of SAE is forward secrecy (*22.9.2 Forward Secrecy*). This means that if an attacker obtains the encryption key for one session, it will not help decrypt older (or newer) sessions. In fact, even if an attacker obtains the master password, it will not be able to obtain any session keys (although the attacker *will* be able to connect to the network). Under WPA2, if an attacker obtains the PMK, then all session keys can be calculated from the nonce values exchanged in the four-way handshake.

As with WPA2, WPA3 requires that both the station and the access point maintain the password cleartext (or at least the key derived from the password). Because each side must authenticate to the other, it is hard to see how this could be otherwise.

WPA3 encrypts even connections to "open" access points, through what is called Opportunistic Wireless Encryption; see **RFC 8110**. WPA3 also introduces longer key lengths, and adds some new ciphers.

Although it is not strictly part of WPA3, the EasyConnect feature was announced at the same time. This allows easier connection of devices that lack screens or keyboards, which makes entering a password difficult. The EasyConnect device should come with a QR code; scanning the code allows the device to be connected.

Finally, WPA3 contains an official fix to the KRACK attack.

3.7.5.4 Encryption Coverage

Originally, encryption covered only the data packets. A common attack involved forging management packets, *eg* to force stations to disassociate from their access point. Sometimes this was done continuously as a denial-of-service attack; it might also be done to force a station to reassociate and thus reveal a hidden SSID, or to reveal key information to enable a brute-force decryption attack.

The 2009 IEEE 802.11w amendment introduced the option for a station and access point to negotiate **management frame protection**, which encrypts (and digitally signs) essential management packets exchanged *after* the authentication phase is completed. This includes those station-to-access-point packets requesting deauthentication or disassociation, effectively preventing the above attacks. However, management frame protection is (as of 2015) seldom enabled by default by consumer-grade Wi-Fi access points, even when data encryption is in effect.

3.7.6 Wi-Fi Monitoring

Again depending on ones driver, it is sometimes possible to monitor all Wi-Fi traffic on a given channel. Special tools exist for this, including aircrack-ng and kismet, but often plain WireShark will suffice if one can get the underlying driver into so-called "monitor" mode. On Linux systems the command `iwconfig wlan0 mode monitor` should do this (where `wlan0` is the name of the wireless network interface). It may be necessary to first kill other processes that have the `wlan0` interface open, *eg* with `service NetworkManager stop`. It may also be necessary to bring the interface down, with `ifconfig wlan0 down`, in which case the interface needs to be brought back `up` after entering monitor mode. Finally, the receive channel can be set with, *eg*, `iwconfig wlan0 channel 6`. (On some systems the interface name may change after the transition to monitor mode.)

After the mode and channel are set, Wireshark will report the 802.11 management-frame headers, and also the so-called radiotap header containing information about the transmission data rate, channel, and received signal strength.

One useful experiment is to begin monitoring and then to power up a Wi-Fi enabled device. The WireShark display filter `wlan.addr == device-MAC-address` helps focus on the relevant packets(or, better yet, the capture filter `ether host device-MAC-address`). The WireShark screenshot below is an example.

```
 7 0.084938505   SamsungE_03:ef:ad   Broadcast           802.11    158 Probe Request, SN=23, FN=0, Flags=........C
 8 0.086005244   Cisco-Li_d1:24:40   SamsungE_03:ef:ad   802.11    139 Probe Response, SN=1230, FN=0, Flags=......
 9 0.115989487   SamsungE_03:ef:ad   Cisco-Li_d1:24:40   802.11     75 Authentication, SN=24, FN=0, Flags=........
10 0.116954409   Cisco-Li_d1:24:40   SamsungE_03:ef:ad   802.11     72 Authentication, SN=1231, FN=0, Flags=......
11 0.118707312   SamsungE_03:ef:ad   Cisco-Li_d1:24:40   802.11    131 Association Request, SN=25, FN=0, Flags=...
12 0.119764782   Cisco-Li_d1:24:40   SamsungE_03:ef:ad   802.11     88 Association Response, SN=1232, FN=0, Flags=
13 0.120082784   Cisco-Li_d1:24:40   SamsungE_03:ef:ad   EAPOL     165 Key (Message 1 of 4)
14 0.148433526   SamsungE_03:ef:ad   Cisco-Li_d1:24:40   EAPOL     189 Key (Message 2 of 4)
15 0.151498719   Cisco-Li_d1:24:40   SamsungE_03:ef:ad   EAPOL     191 Key (Message 3 of 4)
16 0.154637132   SamsungE_03:ef:ad   Cisco-Li_d1:24:40   EAPOL     165 Key (Message 4 of 4)
```

we see node SamsungE_03:3f:ad broadcast a probe request, which is answered by the access point Cisco-Li_d1:24:40. The next two packets represent the open-authentication process, followed by two packets representing the association process. The last four packets, of type `EAPOL`, represent the WPA2-Personal four-way authentication handshake.

3.7.7 Wi-Fi Polling Mode

Wi-Fi also includes a "polled" mechanism, where one station (the Access Point) determines which stations are allowed to send. While it is not often used, it has the potential to greatly reduce collisions, or even eliminate them entirely. This mechanism is known as "Point Coordination Function", or PCF, versus the collision-oriented mechanism which is then known as "Distributed Coordination Function". The PCF name refers to the fact that in this mode it is the Access Point that is in charge of coordinating which stations get to send when.

The PCF option offers the potential for regular traffic to receive improved throughput due to fewer collisions. However, it is often seen as intended for real-time Wi-Fi traffic, such as voice calls over Wi-Fi.

The idea behind PCF is to schedule, at regular intervals, a contention-free period, or **CFP**. During this period, the Access Point may

- send Data packets to any receiver
- send **Poll** packets to any receiver, allowing that receiver to reply with its own data packet

- send a combination of the two above (not necessarily to the same receiver)
- send management packets, including a special packet marking the end of the CFP

None of these operations can result in a collision (unless an unrelated but overlapping Wi-Fi domain is involved).

Stations receiving data from the Access Point send the usual ACK after a SIFS interval. A data packet from the Access Point addressed to station B may also carry, piggybacked in the Wi-Fi header, a Poll request to another station C; this saves a transmission. Polled stations that send data will receive an ACK from the Access Point; this ACK may be combined in the same packet with the Poll request to the next station.

At the end of the CFP, the regular "contention period" or CP resumes, with the usual CSMA/CA strategy. The time interval between the start times of consecutive CFP periods is typically 100 ms, short enough to allow some real-time traffic to be supported.

During the CFP, all stations normally wait only the Short IFS, SIFS, between transmissions. This works because normally there is only one station designated to respond: the Access Point or the polled station. However, if a station is polled and has nothing to send, the Access Point waits for time interval **PIFS** (PCF Inter-Frame Spacing), of length midway between SIFS and IFS above (our previous IFS should now really be known as DIFS, for DCF IFS). At the expiration of the PIFS, any non-Access-Point station that happens to be unaware of the CFP will continue to wait the full DIFS, and thus will not transmit. An example of such a CFP-unaware station might be one that is part of an entirely different but overlapping Wi-Fi network.

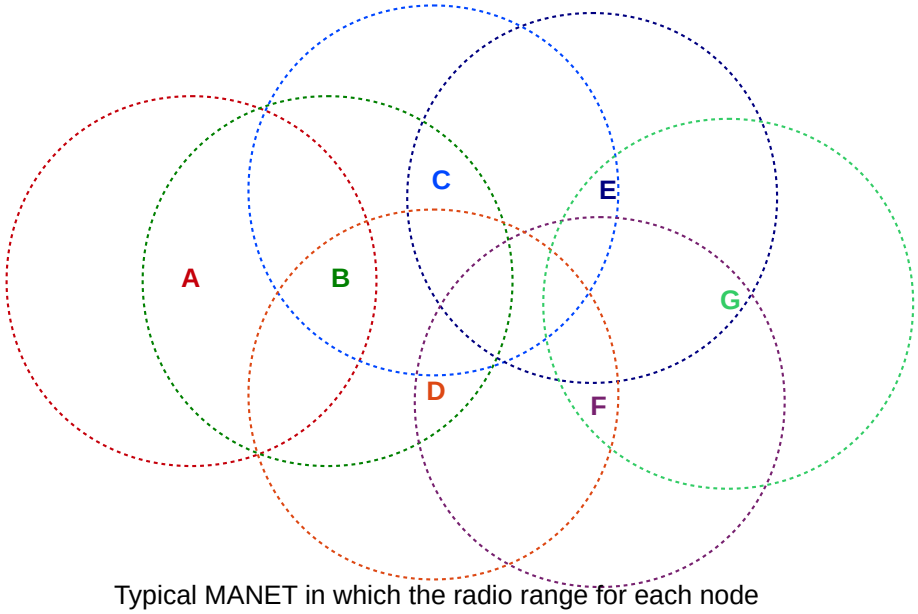
The Access Point generally maintains a **polling list** of stations that wish to be polled during the CFP. Stations request inclusion on this list by an indication when they associate or (more likely) reassociate to the Access Point. A polled station with nothing to send simply remains quiet.

PCF mode is not supported by many lower-end Wi-Fi routers, and often goes unused even when it is available. Note that PCF mode is collision-free, *so long as no other Wi-Fi access points are active and within range*. While the standard has some provisions for attempting to deal with the presence of other Wi-Fi networks, these provisions are somewhat imperfect; at a minimum, they are not always supported by other access points. The end result is that polling is not quite as useful as it might be.

3.7.8 MANETs

The MANET acronym stands for **mobile ad hoc network**; in practice, the term generally applies to ad hoc wireless networks of sufficient complexity that some internal routing mechanism is needed to enable full connectivity. A mesh network in the sense of *3.7.4.4 Mesh Networks* qualifies as a MANET, though MANETs also include networks with much less centralized control, and in which the routing nodes may be highly mobile. MANETs are also potentially much larger, with some designs intended to handle many hundreds of routing nodes, while a typical Wi-Fi mesh network may have only a handful of access points. While MANETs be built with any wireless mechanism, we will assume here that Wi-Fi is used.

MANET nodes communicate by radio signals with a finite range, as in the diagram below.



Typical MANET in which the radio range for each node is represented by a circle around that node. A can reach G either by the route A—B—C—E—G or by A—B—D—F—G.

Each node's radio range is represented by a circle centered about that node. In general, two MANET nodes may be able to communicate only by relaying packets through intermediate nodes, as is the case for nodes A and G in the diagram above. Finding the optimal route through those intermediate nodes is a significant problem.

MANETs for the People

In the early years of MANETs, many designs focused on a decentralized, communitarian approach, *eg* wireless community networks. During the 2010 Arab Spring, MANETs were often proposed (in conjunction with a few users having satellite-Internet access) as a way to bypass government censorship of the Internet. Fast forward to 2018, and much press discussion of "mesh networks" is oriented towards those with exceptionally large private residences. Nothing endures but change.

In the field, the radio range of each node may not be very circular at all, due to among other things signal reflection and blocking from obstructions. An additional complication arises when the nodes (or even just obstructions) are moving in real time (hence the "mobile" of MANET); this means that a working route may stop working a short time later. For this reason, and others, routing within MANETs is a good deal more complex than routing in an Ethernet. A switched Ethernet, for example, is required to be loop-free, so there is never a choice among multiple alternative routes.

Note that, without successful LAN-layer routing, a MANET does not have full node-to-node connectivity and thus does not meet the definition of a LAN given in *1.9 LANs and Ethernet*. With either LAN-layer or IP-layer routing, one or more MANET nodes may serve as gateways to the Internet.

Note also that MANETs in general do not support broadcast or multicast, unless the forwarding of broadcast and multicast messages throughout the MANET is built in to the routing mechanism. This can complicate the operation of IPv4 and IPv6 networks, even assuming that the MANET routing mechanism replaces the

need for broadcast/multicast protocols like IPv4's ARP (*7.9 Address Resolution Protocol: ARP*) and IPv6's Neighbor Discovery (*8.6 Neighbor Discovery*) that otherwise play important roles in local packet delivery. For example, the common IPv4 address-assignment mechanism we will describe in *7.10 Dynamic Host Configuration Protocol (DHCP)* relies on broadcast and so often needs adaptation. Similarly, IPv6 relies on multicast for several ancillary services, including address assignment (*8.7.3 DHCPv6*) and duplicate address detection (*8.7.1 Duplicate Address Detection*).

MANETs are simplest when all the nodes are under common, coordinated management, as in the mesh Wi-Fi described above. Life is much more complicated when nodes are individually owned, and each owner wishes to place limits on the amount of "transit traffic" – traffic passing through the owner's node – that is acceptable. Yet this is often the situation faced by schemes to offer Wi-Fi-based community Internet access.

Finally, we observe that while MANETs are of great theoretical interest, their practical impact has been modest; they are almost unknown, for example, in corporate environments, beyond the mesh networks of *3.7.4.4 Mesh Networks*. They appear most useful in emergency situations, rural settings, and settings where the conventional infrastructure network has failed or been disabled.

3.7.8.1 Routing in MANETs

Routing in MANETs can be done either at the LAN layer, using physical addresses, or at the IP layer with some minor bending (below) of the rules.

Either way, nodes must find out about the existence of other nodes, and appropriate routes must then be selected. Route selection can use any of the mechanisms we describe later in *9 Routing-Update Algorithms*.

Routing at the LAN layer is much like routing by Ethernet switches; each node will construct an appropriate forwarding table. Unlike Ethernet, however, there may be multiple paths to a destination, direct connectivity between any particular pair of nodes may come and go, and negotiation may be required even to determine which MANET nodes will serve as forwarders.

Routing at the IP layer involves the same issues, but at least IP-layer routing-update algorithms have always been able to handle multiple paths. There are some minor issues, however. When we initially presented IP forwarding in *1.10 IP - Internet Protocol*, we assumed that routers made their decisions by looking only at the network prefix of the address; if another node had the same network prefix it was assumed to be reachable directly via the LAN. This model usually fails badly in MANETs, where direct reachability has nothing to do with addresses. At least within the MANET, then, a modified forwarding algorithm must be used where *every* address is looked up in the forwarding table. One simple way to implement this is to have the forwarding tables contain only **host-specific** entries as were discussed in *3.1 Virtual Private Networks*.

Multiple routing algorithms have been proposed for MANETs. Performance of a given algorithm may depend on the following factors:

- The size of the network
- How many nodes have agreed to serve as routers
- The degree of node mobility, especially of routing-node mobility if applicable
- Whether the nodes (especially routing nodes) are under common administration, and thus may agree to defer their own transmission interests for the common good
- per-node storage and power availability