Operating Systems

Networking

By Paul Krzyzanowski

April 19, 2012

Everything lives, moves, everything corresponds; the magnetic rays, emanating either from myself or from others, cross the limitless chain of created things unimpeded; it is a transparent network that covers the world, and its slender threads communicate themselves by degrees to the planets and stars. Captive now upon earth, I commune with the chorus of the stars who share in my joys and sorrows.
—Gérard de Nerval, Aurélia, Part 2, chapter 6

Introduction

As we add more processors to a machine, programs and operating systems do not become substantially different. In a symmetric multiprocessor (SMP) architecture, each processor has access to all of system memory and they all run under the same operating system. The scheduler gets the additional responsibility of managing multiple processes in the *running* state and assigning processes to processors. A process can use any existing mechanisms for communicating with other processes, such as shared memory, semaphores, and files.

Once we move beyond standalone systems and consider at a collection of independent computers, we can no longer use these local operating system-based communication mechanisms. To work together, the computers will have to communicate with each other via an inter-machine interconnect: the network.

Some networking terminology

This section will provide a quick tour of some of the more commonly encountered terms encountered in networking.

Most computers are connected to a **local area network**. This is a data communications network that generally covers a small area, such as a home, school, or a small set of buildings. It shares the same transmission medium (e.g., ethernet) and typically has a high data rate and low latency when compared to **wide area networks**. All systems on the network are peers, meaning that no one device is in charge of brokering communications: any device can initiate a data transfer with any other device. Devices (**endpoints**) connected to a LAN are called **nodes**. Most often, nodes on a LAN are **workstations**. This refers to any device that is generally used by a single user at a time and includes most personal computing devices (PCs, Macs, tablets).

For a node to be connected to the LAN, interface hardware is needed. This is known as an **adapter** and is a circuit that is either built onto the main circuit board or sits on an expansion slot/connector. Networking adapters are referred to as **Network Interface Cards**, or **NICs** (even if they are not cards).

Media refers to the wires (or wireless RF) connecting together the devices that make up a LAN. The following types of media are generally encountered:

- **Twisted Pair** is the most common wired medium. It typically has eight wires and comes in two flavors: shielded twisted pair (**STP**) or the more common unshielded twisted pair (**UTP**). Telephone cable is an example of UTP.
- **Coaxial cable** (coax) comes in two flavors as well. Thin coax (similar to TV cable) is by far the more popular of the two. Thick coax is now obsolete. Thin coax is rarely seen in LANs as well but is in widespread use for cable TV/internet service to the home.
- **Fiber** (optical cable) is rarely use for LANS but is common in wide area networks due to its ability to deliver signals for longer distances and higher bandwidth.

- **Wireless**, the most popular of which is the 802.11 family of transceivers (Wi-Fi). These seamlessly extend a wired ethernet LAN to a wireless network.

A **hub** is a device that acts as a central point for LAN cable. It takes incoming data from one cable and broadcasts it to all connected cables. Hubs are rarely used, having been replaced by switches. A **switch** analyzes packet data to determine the the appropriate destination port (or ports) on which to transmit it. This reduces overall congestion. **Concentrators** or **repeaters** regenerate data signals when data passes through them, allowing data to pass through longer distances. **Bridges** connect different LAN segments together to form a larger logical local area network (layer 2; see the later discussion). **Routers** determine the next network point to which packets should be forwarded – they connect different types of local and wide area networks (layer 3).

Modes of connection

There are several ways in which a node can communicate with another node. One way is to use a dedicated connection between the two nodes. This is a **physical circuit** and electrons flow from the NIC on one node directly to the NIC on another node. The old public telephone network operated in this manner. An operator would use a patch panel and physically connect a wire between you and the party you wanted to talk to. This then evolved into an electromechanical system with the invention of the crossbar switch.

Another approach is to have each communicating set of nodes talk on different frequencies (or, more often, tune to the frequency of the transmitter whose data you want to acquire). This is known as **broadband**. With this type of networking, nodes can all communicate at the same time but have to stay in their "channels", or frequency bands. The radio frequency spectrum is allocated into many discrete bands, each of which is reserved for specific functions. For example, TV channels 38-51 are allocated to the range of frequencies between 614 and 698 MHz. Each TV channel is allotted a 6 MHz spectrum. A wall chart of United States frequency allocations can be found here (http://www.ntia.doc.gov/files /ntia/publications/spectrum_wall_chart_aug2011.pdf).

Finally, one can take turns on a shared communication medium. This is called **baseband** communication. Here, a node can use the entire bandwidth of the media; it just cannot use it all the time. There are two ways of sharing the network:

Circuit switching

We can divide the network into short, fixed-length time slots. Each node will be allowed use of specific time slots. This is known as TDMA (Time Division Multiple Access). It is an example of **circuit switching** since we are creating something that has essentially the equivalent performance to an isolated, directly connected network. Because time slots are fixed, the bandwidth that a node gets is predictable, even though it is less than the bandwidth if the node had exclusive use of the medium. The modern (post-crossbar) public switched telephone network is an example of circuit switching. The downside of circuit switching is that it does not lead to efficient use of the network. If a node is not transmitting data, its time slots go unused; if it has a lot of data to transmit, it must wait for its allotted time slots.

In general, with circuit switching, we expect to get a dedicated, pre-defined path from the source to the destination, a fixed bandwidth, and constant latency (rate of data arrival).

Packet switching

The other approach in baseband networking is to share the network by using variable size time slots. A variable-size chunk of data on the network is called a **packet** and we rely on statistical multiplexing for network access. A node has to wait for the network medium to be free of data before it can transmit. Then it sends whatever it needs to transmit: a large or a small packet. This mode of operation is called **packet switching** and is the dominant means of sharing a data network. It also has the advantage that it permits many-to-many communication as a packet can be targeted to multiple nodes (or broadcast to all) concurrently. Since there is no way of identifying one node's packet from another's on the network, each packet must contain a destination address to identify the recipient of the packet.

With packet switching, the bandwidth that a node experiences is based on the amount of traffic that other nodes generate. Hence, we can expect to see a variable bandwidth and variable latency in our communication session.

Ethernet

Ethernet is the most common local area networking technology. It was developed in the mid 1970s at Xerox PARC and standardized by the IEEE 802.3 committee. It is a **baseband transmission** network. This means that all nodes share access to the network media on an equal basis. Data uses the *entire bandwidth* of the media. This is opposed to **broadband transmission**, where a given data transmission uses only a segment of the media by dividing the media into channels (e.g., frequency bands). The typical speed of transmission on an Ethernet network using unshielded twisted pair is currently 1 Gbps, with speeds going up to 10 Gbps and higher (40 and 100 Gbps over fiber).

Ethernet was designed for a shared transmission medium. On a shared channel, the mechanism it uses to get packets onto the network is called **Carrier Sense Multiple Access with Collision Detection** (**CSMA/CD**). To send data, a node first listens to the network to see if it is busy (i.e., someone else is sending data). When the network is not busy, the node will send data and then sense to see whether a collision occurred because some other node decided to transmit data concurrently. If a collision was detected, the data is retransmitted. The analogy of CSMA/CD is that of using a telephone on a shared line. CSMA/CD is not needed on switched ethernet connections.

Along with ethernet, the 802.11 family of protocols (Wi-Fi) for wireless networking enjoys great popularity and is often bridged to be part of the same LAN as an ethernet network. Wi-Fi uses a shared communication medium as well: specific frequency bands in the 2.4 GHz and 5 GHz range. The key difference for NIC to send packets onto a Wi-Fi network as opposed to an ethernet network is that one cannot listen for collisions while sending. For this reason, even though CSMA/CD would have been desirable, it cannot be used. Instead the network is accessed via a scheme known as **CSMA/CA**: Carrier Sense Multiple Access with Collision Avoidance. As with CSMA/CD, the network transceiver on the node first listens on the desired channel to see if it is idle. If it is, then it sends a packet. Otherwise, it waits until transmission stops and adds on a random amount of time to avoid contention with other nodes that might want to transmit on that channel. It then listens again. If the channel is still idle, the packet is transmitted. Since there is no way of knowing if it was actually sent without a collision, the transceiver relies on receiving an acknowledgement from the receiver (e.g., the access point).

Cable TV broadband?

This is a bit outside our scope of discussion but is worth mentioning to allay any confusion. When cable TV companies began offering Internet service to compete with dial-up modems, their marketing departments jumped on the term "broadband" and it has since taken on a meaning of "fast Internet service". Unfortunately, the use of the term is not quite correct in the engineering sense. Cable TV is indeed a broadband service: each TV channel is allocated a distinct set of frequencies (6 MHz for HDTV and analog channels; several SD digital can fit into that spectrum). However, Internet service is provided by taking one or more of these frequency bands and using it as a packet-switched *shared* data network. The specifications are defined by DOCSIS (Data Over Cable Services Interface Specification). A DOCSIS modem tunes in the appropriate band to demodulate the data signal and send it to an ethernet (and/or Wi-Fi) interface. In essence, Cable TV Internet service is baseband service that is delivered as one data flow on a broadband network.

Parlez-vous français? ¡Sí, muy bien!

For computers (or people, for that matter) to be able to communicate, they must speak the same language and follow the same conventions. For humans, this means speaking the same language and knowing how low to bow, which hand gestures not to use, and whether it is acceptable to excrete gas. For computers, this requires knowing how to find out how long a packet is that is coming over a network (so we can get it), knowing where the destination address is stored, and being able to properly interpret all the data within it. The issue is non-trivial because different computers have different concepts of which order to store bytes of a word, how long an integer is, and what character set is being used. The instructions and conventions needed for successful communication is known as a protocol. The instructions and conventions for making sense of data and for computing are known as **protocols**.

To ease the task of communicating and provide a degree of flexibility, network protocols are generally organized in **layers**. This allows one to replace a layer of the protocol without having to replace the surrounding layers. It saves higher-level software from having to bother with formatting an ethernet packet. The most popular model of guiding (not specifying) protocol layering is the OSI Reference Model, designed in 1977 and refined somewhat thereafter. It contains seven layers of protocols:

1. Physical

Deals with the specification of the data signals, voltage levels, transmission speed, and connectors. Examples include an RS-232 serial connector or a 1000BASE-T gigabit ethernet connector.

2. Data link

Provides the first level of organization of data – the datalink frame (packet), which includes the source address, destination address, content, and some form of checksum for error detection. This layer includes Media Access Control (MAC) and Logical Link Control (LLC). MAC covers rules for accessing the media and dealing with contention. The LLC portion covers frame synchronization, flow control, and error checking. Examples of this layer include Ethernet data (MAC-layer) and PPP (the Point-to-Point Protocol).

3. Network

Relay and route information to the destination node. This layer is responsible for managing the journey of packets both within and across local area networks and figuring out intermediate hops (if needed). The network layer gives us the full abstraction of machine-to-machine communication. Examples of this layer include the Internet Protocol (IP) and X.25.

4. Transport

Provides reliable end-to-end communications by providing service-level (transport) addressing, flow control, datagram segmentation, and end-to-end error checking. It can ensure that packets appear to arrive in the correct order and issue retransmission requests to ensure the reliable message delivery. The network layer gives us the abstraction of application-to-application communication. Examples of this layer include TCP/IP and UDP/IP: two transport-layer protocols over the Internet Protocol (IP).

5. Session

Responsible for connection establishment, data transfer, and for connection release. It tracks who initiated a conversation and may manage the re-establishing of a logical communication channel. Examples of this include HTTP 1.1 and SSL (Secure Socket Layer, aka TLS).

6. Presentation

Responsible for the selection of an agreed-upon syntax (data representation). This layer may have to convert data between the agreed-upon representation and the machine's native types. Examples of this include MIME, XDR (the eXternal Data Representation used by ONC RPC), Google Protocol Buffers, and ASN.1 (Abstract Syntax Notation).

7. Application

The protocol of applications using networking, such as file transfer, directory services, distributed processing applications, and many others. Examples of this include email (SMTP, POP, and IMAP protocols), file transfer (FTP), and directory services (LDAP).

Client-Server communication

The most common networking relationship is the **client-server model**. The model contains two communicating processes: a **client** running on a client node and a **service** running on a **server**. A **service** is that task that a machine can perform, such as retrieving files over a network, presenting web page content, or the executing a command on the node. A **server** is the machine that performs the task (the machine that offers the service). These titles are generally used in the context of a particular service rather than in labeling a machine: one machine's client may be another machine's server.

To offer a service, a server must get a **transport address** for a particular service. This is a well-defined location (similar to a telephone number) that will serve to identify the service. The server associates the service with this address before clients can communicate with it. It is important to distinguish a *transport address* from a *node (or machine) address*. A machine address allows us to send messages to a node but the system will have no clue as to the disposition of the data: which application it is targeted for. A transport address operates at layer four of the OSI stack and allows one to identify a communications endpoint within the node.

The client, wishing to obtain a service from the server, must obtain the server's transport address. There are several ways to do this: it may be hard-coded in an application or it may be found by consulting a database (similar to finding a number in a phone book). The database may be as simple as a single file on a machine (e.g. /etc/services on Unix systems) or as complex as accessing a distributed directory server.

We depend on transport providers to transmit data between applications. A **transport provider** is a piece of software that accepts a network message and sends it to a process on a remote machine. There are two categories of transport protocols:

connection-oriented protocols

These are analogous to placing a phone call:

- first, you establish a connection (dial a phone number)
- possibly negotiate a protocol (decide which language to use)
- communicate
- terminate the connection (hang up)

This form of transport is known as **virtual circuit service** since it provides the illusion of having a dedicated circuit. Messages are guaranteed to arrive in order. It is virtual because it does not change the physical network into a circuit-switched network but just simulates its desirable property of in-order delivery, reliable delivery, and keeping state of where packets are going. It cannot do anything about latency or bandwidth.

connectionless protocols

These are analogous to sending mail:

- there is no connection setup
- data is transmitted when ready (drop a letter in the mailbox)
- there's no termination because there was no call setup

This transport is known as **datagram service**. With this service, the client does not know whether the message arrived at the destination or whether the data arrived in the same order that it was transmitted. Datagram service is less reliable than virtual circuit service but has less overhead.

Ethernet

Ethernet is currently the most common local area network. Ethernet communication represents layers one and two of the OSI model. Layer 1 covers the physical aspects of ethernet connectivity. This includes the 8P8C (RJ45) connector, 1000BASE-T cable, and the voltage levels. Layer 2 covers the Data Link layer and includes error detection, data frame transmission (e.g., implementing the CSMA/CD logic), data frame parsing, and ethernet bridging. Altogether, ethernet offers unreliable connectionless communication over a local area network.

Ethernet has no relationship to IP or, for that matter, other networks. It is a packet-based network that identifies a destination with a 48-bit ethernet address. The packet size is variable-length and is limited (MTU, maximum transfer unit) to 1,518 bytes that consist of an 18-byte header and up to 1500 bytes of data. With the advent of gigabit ethernet, support was added for "jumbo packets", which support a 9,000 byte MTU. Ethernet hardware on each node is constantly monitoring the network to detect packets where the destination address matches its own address. If a match is found, the packet contents are copied to an internal hardware buffer and an interrupt is generated to inform the operating system that incoming data is ready.

IP: Internet Protocol

By far the most popular network protocol these days is the family of **Internet Protocols**. The Internet was born in 1969 as a research network of four machines that was funded by the Department of Defense's Advanced Research Projects Agency (ARPA). The goal was to build an efficient, decentralized, fault-tolerant network that could connect heterogeneous machines and link together separately connected networks. The network protocol is called the **Internet Protocol**, or **IP**. It is a connectionless protocol that is designed to handle the interconnection of a large number of local and wide area networks that comprise the Internet.

IP may route an IP packet from one physical network to another. Every machine on an IP network is assigned a unique 32-bit IP address. When an application sends data to a machine, it must address it with the IP address of that machine. The IP address is not the same as the machine address (e.g. the ethernet address) but is strictly a logical address. There is no relationship between the logical IP address and the hardware address on the underlying physical network. If a machine is connected to several physical networks, it will have several IP addresses, one for each network.

IP addressing

A 32-bit address can potentially support 2^{32}, or 4,294,967,296 addresses. However, if every machine on an IP network would receive an arbitrary IP address, then routers would need to keep a table of over four billion entries to know how to direct traffic throughout the Internet! To deal with this more sensibly, routing tables were designed so that one entry can match multiple addresses. To do this, a **hierarchy** of addressing was created so that machines that are physically close together (say, in the same organization) would share a common prefix of bits in the address. For instance, consider the two machines:

name	*address*	*hex address*
cs.rutgers.edu	128.6.4.2	`80 06 04 02`
sakai.rutgers.edu	128.6.31.88	`80 06 1f 58`

The first sixteen bits identify the entire set of machines within Rutgers University. Systems outside of Rutgers that encounter any destination IP address that begins with 0x8006 have only to know how to route those packets to some node (a router) within Rutgers that can take care of routing the exact address to the proper machine. This saves the outside world from keeping track of up to 65,536 (2^{16})machines within Rutgers. An IP address is segmented into two parts:

- **network number** — identifies the network that the machine belongs to
- **host number** — identifies a machine on that network.

The network number is used to route the IP packet to the correct local area network. The host number is used to identify a specific machine once in that local area network. If we use a fixed 16-bit partition between network numbers and host numbers, we can have have a maximum of 65,536 (2^{16}) separate networks on the Internet, each with a maximum of 65, 536 hosts. The expectation, however, was that there would be a few big networks and many small ones. To support this, networks are divided into several classes. These classes allow the address space to be partitioned into a few big networks that can support many machines and many smaller networks that can support few machines. With class-based addressing, The first bits of an IP address identify the class of the network.

class	*leading bits*	*bits for network number*	*bits for network number*
A	`0`	7	24
B	`10`	14	16
C	`110`	21	8

An IP address is usually written as a sequence of four bytes, each byte in decimal, separated by periods. For example, an IP address written as 135.250.68.43 translates into the hexadecimal address 87FA442B (135=0x87, 250=0xfa, etc.). In binary, this address is `1000 0111 1111 1010 0100 0100 0010 1011`. The leading bits of this address are 10, which identifies the address as belonging to a class B network. The next 14 bits (00 0111 1111 1010) contain the network number (7FA) and the last 16 bits contain the host number (442B).

To allow organizations to create additional networks without requesting additional (and increasingly scarce) network numbers, some high bits of the host number may be allocated for a network number within a higher-level IP network. These local networks are known as **subnets**. Routers within an organization can be configured to extract this additional network ID and use it for routing. For example, a standard class B network allows 16 bits for a host number. This host address may be locally broken into 8 bits for a subnet ID followed by 8 bits for a host ID.

Machines in an IP network are named in a hierarchical manner, with each naming level separated by a dot. Names to the left are lower in the hierarchy. For example, the name `bescot.cl.cam.ac.uk` identifies a machine named `bescot` in England (`uk`), under the Academia hierarchy (`ac`), within Cambridge University (`cam`), within the Computer Laboratory (`cl`). There is no relationship whatsoever between the hierarchy of naming machines and the underlying IP addresses. An IP address corresponding to a name can found by looking it up in some database. In the past (before the mid 1980s), that database was a single file (`/etc/hosts`) that contained the address of every machine on the Internet. As the Internet grew, that file became difficult to manage. Now, in most cases, you would contact a network-based `name service` offered by some machine that may in turn contact (or tell you to contact) other name servers until it finds a machine that knows the address for a given name. These name servers are known as `Domain Name Servers`, or `DNS`.

Running out of IP addresses

As the Internet expanded in the early 1990s to include more networks and more hosts, the three-layer class hierarchy of IP addresses was getting stressed. More and more organizations wanted to be on the Internet. To be on the Internet, each machine would need an IP address. An organization would request an IP network address for a class that was sufficiently large to accommodate all of its machines (and all of the machines it plans to get in the future).

The problem we were having with class-based Internet addressing was not in running out of IP addresses (exhausting all possible IP addresses) but rather running out of IP network addresses (of which there are only two million available). Many organizations also wanted more than a class C network and there are only a bit over 16,000 class A and B networks available.

While the class-based IP addressing scheme can accommodate close to four million distinct addresses, the problem was that the class-based network granularity was too coarse. For example, a class C network can support up to 254 host numbers. If an organization needed up to 1,000 hosts, it would need to request a far more precious class B network (of which there are only 16,382 such networks available). The class B network will allow it to have up to 65,534 host numbers, meaning that over 64,000 IP addresses, or over 98% of the address space, will go unused.

To combat this problem, a routing structure called **Classless Inter-Domain Routing** (**CIDR**) was created. This is a structure that tries to provide a better match between the range of addresses assigned to an organization and the number of addresses the organization really needs.

The practice of identifying a class of network (A, B, or C) by looking at the leading bits of the IP address was abandoned. Instead, an IP network number is defined to be a n arbitrary number of leading bits of an address. Using the earlier example, if an organization needed to support 1,000 hosts, it would request a class B address, wasting over 64,000 addresses. Now it can request a 22-bit network number, which provides it with 10 bits of addressing for hosts, enough for 1,022 machines.

Since we can no longer look at the leading bits of an IP address and know how many of the following bits constitute the network number, each entry in the routing table now has to contain this number explicitly. A CIDR IP address includes the standard 32-bit IP addresses and information on the number of bits for the network prefix (for example, 128.6.13.3/16, where the 16 refers to a sixteen bit network number). The pitfall of CIDR is that one now has to manage the prefixes as well as addresses. When routing tables are distributed, they must include the prefixes for the addresses to make sense from a routing point of view.

CIDR requires router tables contain both an IP address and the number of bits for the network prefix. This structure helps alleviate another problem in IP networking: large global routing tables. With class-based IP addressing, every network had to have a routing entry in global routers on the Internet. This was both an administrative pain and a performance bottleneck. If network addresses can be assigned "sensibly", routing tables can be simplified. If adjacent network addresses are generally routed in the same way (for example, they belong to the same ISP and the routes split up only when they get to that ISP’s network), then the global routing tables do not need to contain all those networks; they can simply specify that less bits are significant for the route (i.e., as far as the router is concerned it is a single route to the network with less bits being used for the network number).

Another innovation in IP addressing also helped alleviate the problem of assigning network addresses to organizations. The idea is that if every machine in an organization does not need to be addressed from the Internet, it need not have a unique IP address. Machines within an organization now can have internal IP addresses that are not unique across the Internet but only unique within the organization. Whenever a machine sends a packet outside the organization, it is routed through a gateway that will translate the address from an internal address to an external address. This will be the address of the gateway system (which must have a true external IP address). In translating the address, the gateway will keep a table of the original address and port number and the outgoing, translated, port number. When a return packet comes for that port number, the gateway identifies it as a response to the original packet and rewrites the destination in the IP header as the internal address and port number of the original sender. This scheme is known as **network address translation**, or **NAT**. The biggest benefit of NAT is that large organizations no longer need to request a network that can address thousands or tens of thousands of hosts. They only need to support the number of hosts that need to be visible from the Internet (e.g., those running services) as well as gateways.

All these approaches were designed to buy us time for the inevitable future when we really will run out of IP addresses. To combat this problem, a new version of the IP protocol was designed, called IPv6: IP version 6, to distinguish it from the predecessor, IP version 4. With IPv6, addresses are 128 bits instead of 32 bits. Varying number of leading bits identify (http://www.iana.org/assignments/ipv6-address-space/ipv6-address-space.xml) various categories of addresses. For example the top three bits may identify a global unicast address space (http://tools.ietf.org/html/rfc4291). In general 48 bits are used to identify a "routing prefix" (network) and 64 bits are used to identify a network interface on a node. Given that there are approximately 1.33×10^{50} atoms on Earth, it is extremely unlikely that we will run out of addresses.

IP Routing

How does a packet find its way from here to there? A switching element is used to connect two or more transmission lines (e.g., Ethernet networks). This switching element is known as a **router**. It can be a dedicated piece of hardware or a general-purpose computer with multiple network interfaces. When it gets packet data, it has to decide to which line the data has to be sent. This job is called **routing**. When a router receives an IP packet, it checks the destination address. If the destination address matches that of the receiving system, then the packet is delivered locally. Otherwise, router uses the destination address to search a **routing table**. Each entry in the table has an address, the number of significant bits (usually represented by a bit mask known as a netmask), and an outgoing interface. When an entry is located that matches the IP destination address (not counting the bits outside the netmask), the packet can be sent out on the interface defined on that line. This technique is known as **static routing**. An alternative to static routing is **dynamic routing**, which is a class of protocols by which machines can adjust routing tables to benefit from load changes and failures.

Drivers

We will cover the driver and data flow structure in more detail when we look at sockets in the next lecture.

IP driver (network protocol)

IP is a logical network that sits on top of multiple physical networks. Operating systems that support communication over IP have module that is called an IP driver. The IP driver is responsible for implementing the Internet Protocol and performs the following operations:

- getting operating parameters from the network device driver (which controls the network card) such as maximum packet size, functions to initialize the hardware headers, and the length of the hardware header
- routing packets from one physical network to another
- fragmenting packets: it might have to send a packet that is too big for the network hardware to handle in which case it has to be split into several packets, each with its own IP header containing destination information
- performing send operations from higher level software
- receiving data from the device driver
- dropping data with bad checksums in the header
- dropping expired packets

Ethernet driver (network device driver)

A logical network has to communicate with an underlying physical network. That network is typically an ethernet network (or Wi-Fi, which is conceptually very similar). The **ethernet driver** is responsible for interfacing to the NIC. It has to:

- Process interrupts from the network transceiver, receive packets, and send them up to the higher levels (the IP driver if the data is an IP packet).

- Get packets from the higher levels (e.g., the IP driver) and send them to the hardware, ensuring that the packet goes out successfully.

Before an IP packet is sent on an ethernet network, it has to be **encapsulated**, or **enveloped**, for the physical (ethernet) network. **Packet encapsulation** means that the entire IP packet is simply treated as data as far as the ethernet network is concerned. The IP packet is placed within an ethernet packet, that must have a valid ethernet address in it.

To get a valid ethernet destination address for an IP address, the IP address is first looked up in the routing table to see if it needs to be routed to a specific IP address within the LAN. If so, that address will be used as the destination IP address instead of the one we had in the packet. The destination IP address is then converted to its corresponding ethernet address via the **Address Resolution Protocol** (**ARP**). ARP finds the corresponding ethernet address via the following steps:

1. check the local ARP cache
2. send a broadcast ethernet packet requesting the ethernet address of a machine with a certain IP address
3. wait for a response (with a time out period)

Note that there is no relationship between the IP address for a network port on a node and that port's ethernet address.

Protocols over IP

IP supports two transport layer protocols: TCP and UDP. There are also special protocols for sending control messages to guide routing decisions that are not used for application-level communication. These include ICMP (Internet Control Message Protocol) and RIP (Router Information Protocol) as well as others, such as OSPF, IGRP, EIGRP, IS-IS, and BGP.

The two transport layer protocols over IP are:

TCP — Transport Control Protocol

- virtual circuit service (connection-oriented)
- sends acknowledgment for each packet received
- checksum to validate data contents
- data may be transmitted simultaneously in both directions over a circuit
- no record markers (one write may have to be read with multiple reads) but data arrives in sequence

UDP — User Datagram Protocol

- datagram service (connectionless)
- data sent may be lost
- data may arrive out of sequence
- recipient’s address must be specified in each request

Applications may use either of these protocols to send data over the network. To identify communication endpoints within a machine (e.g., one application's network connection versus another's), these transport-layer protocols introduce a 16-bit value in their header called a **port number**. For example, you might contact an SMTP mail server on a machine on TCP port 25, while contacting the HTTP web server on the same machine (same IP address) on port 80.

Both TCP and UDP headers are encapsulated within an IP packet. That is, IP just treats those headers as data along with the user's data in the packet. The IP header has a 6-bit *protocol* field that identifies the type of protocol that is encapsulated within it. This allows an

IP driver to send the packet to the right transport-level protocol (e.g., to the TCP module). Likewise, an Ethernet packet (called a "frame") treats the entire IP packet as data. The Ethernet header has a two-byte value that identifies the type of protocol that it is encapsulating so that an ethernet driver can send the data to the correct protocol module (e.g., to the IP driver).

References

- Anatomy of the Linux networking stack (http://www.ibm.com/developerworks/linux/library/l-linux-networking-stack/), IBM developerWorks, M. Tim Jones, Emulex Corp.
- Classless Inter-Domain Routing (CIDR) Overview (http://public.pacbell.net/dedicated/cidr.html), Pacific Bell Internet
- TCP/IP Illustrated (http://www.amazon.com/TCP-IP-Illustrated-3-Set/dp/0201776316/pkorg), Volume 1: The Protocols, W. Richard Stevens, Gary R. Wright, ©2001 Addison-Wesley Professional. *[if you want a great reference on the TCP protocol, this set of books is the one to get]*