How the Internet works

Rolfe Bozier

3-Apr-2014

An introduction to TCP/IP protocols

- What happens when I type "http://www.google.com/" in my browser?
- Why do I care?
  - developing a distributed system
  - debugging a distributed system
  - why can't I connect to a remote service?
  - how would I connect two system together?
  - how can I tell what is happening behind the scenes?

Packets and protocols

- Data is exchanged between computers in "packets"
  - A packet usually comprises a header and a payload
  - Packets can be nested - a payload can contain another packet, with its own header and payload
  - Packet headers have a well-defined structure and content
- A protocol is a definition of a set of packet layouts and how they are exchanged
- Because packets can be nested, a protocol can encapsulate another protocol (and so on)
  - This nesting of protocols allows us to extend a protocol to support more features and functionality
- Packets are good for a number of reasons:
  - easy to manage latency
  - easy to spread / balance load
  - error recovery is more efficient

The Internet runs on the TCP/IP protocol suite

- The Internet is built on a hierarchy of protocols known colloquially as TCP/IP
- We will cover the following in this talk:

Protocol Hierarchy			OSI model
	DNS	HTTP	*???*
	UDP	TCP	*Transport layer?*
ARP	IP		*Network layer*
Ethernet			*Data link layer*

The lowest level (that we will consider)

- I will assume that we have a "data link" layer between your computer and the rest of the world (usually a network switch)
  - typically this is an *ethernet* link
  - or maybe PPP over a dialup or *DSL* line
- Each end has a "link address" (e.g. MAC / hardware / ethernet address)
- There is an electrical connection between the two ends
- Each end can send bytes to the other (maybe at the same time)

Ethernet packets

- Ethernet packets:

src addr	dst addr	type	payload

- src addr is the MAC address of your computer
- dst addr is the MAC address at the other end of the cable (on your local network)
- To send this packet, just send it down the wire (final delivery is not your problem!).
- Ethernet is a broadcast medium.
- Historically:
  - everyone on the same piece of coax (don't break it!)
  - Later, hubs allowed people to connect/disconnect but it was still broadcast
  - Today, bridges/switches partition the network into segments; these devices know which way to pass on a packet
- The address space is still flat

Separating addressing from hardware - ARP

- MAC addresses don't scale or allow partitioning
- IP addresses, on the other hand, are hierarchical: A.B.C.D
  - prefix part is network (e.g. 192.168.20)
  - rest is host (.100)
- Each host is assigned an IP address
- Linking IP addresses to hardware addresses is done by the **ARP** protocol

ether hdr	src ether addr	src IP addr	dst ether addr	dst IP addr

- An ARP request leaves one of the last 2 fields blank and broadcasts it to the entire local network
- A destination that recognises its ether/IP address sends a reply back to the src ether address with the missing details
- The original host now knows the matching ether address for an IP address
  - Replies are cached for a while for efficiency
  - If no reply, try a few times then give up

Direct host-host communications - IP

- Now we can talk with other computers using IP addresses
- But the ethernet header contains the dst ether address, so why do we need a dst IP addr?
  - IP networks can span beyond your local network
  - routers can manage the exchange of packets between networks
- IP addresses are hierarchical, so we can efficiently route networks:
  - send all packets for 203.8.* to *that* router over there
  - send all unknown addresses to your internet router
- We can address another IP host using the IP protocol:

ether hdr	header	src IP addr	dst IP addr	payload

- The IP protocol also handles:
  - fragmentation / reassembly when your data is bigger than the max packet size
  - checksums for error detection
  - time-to-live - expire packets if they travel too far (e.g. in a loop)
  - what protocol is encapsulated within the payload?
- Now we can send a packet to an IP address, but what happens to it there?

Direct host-host communications - IP

- Now we can talk with other computers using IP addresses
- But the ethernet header contains the dst ether address, so why do we need a dst IP addr?
  - IP networks can span beyond your local network
  - routers can manage the exchange of packets between networks
- IP addresses are hierarchical, so we can efficiently route networks:
  - send all packets for 203.8.* to *that* router over there
  - send all unknown addresses to your internet router
- We can address another IP host using the IP protocol:

ether hdr	header	src IP addr	dst IP addr	payload

- The IP protocol also handles:
  - fragmentation / reassembly when your data is bigger than the max packet size
  - checksums for error detection
  - time-to-live - expire packets if they travel too far (e.g. in a loop)
  - what protocol is encapsulated within the payload?

Host and domain names

- IP addresses are fine technically, but not much good for people
  - Host and domain names are a human-friendly hierarchical naming system
  - Matching IP addresses to hostnames is done by the **DNS** protocol
  - DNS is a **UDP**-based protocol on top of IP
- UDP - datagram protocol
  - sometimes you may want to send one packet, maybe with a reply

ether hdr	IP hdr	header	src port	dst port	other	payload

- The dst port controls which service will process the packet at the dst
  - The src port allows for replies
  - UDP is stateless, not guaranteed, unordered
  - But it is useful if you can tolerate lost packets, for broadcasts, or if you don't really care

DNS – domain name system

- Name / address lookup is provided by DNS servers
  - typically you start by querying a local DNS server

ether hdr	IP hdr	UDP hdr	request[s]	reply[s]

- The requestor provides the requests, and server fills in the answers
- DNS uses UDP, so packets may get lost
- The protocol allows for retries and failover to alternate servers
- DNS servers can be very busy, so a lightweight protocol is good
- Now we can talk with other computers using host *names*

Talking to web servers

- A visit to a web page in your browser typically involves the exchange of a lot of different types of data (web pages, style sheets, images, forms etc.)
  - So UDP is not really appropriate
- HTTP is a **TCP**-based protocol on top of IP
- TCP is a protocol that provides connection-based communications
  - A TCP connection is a long-lived conversation between two hosts with two-way traffic

ether hdr	IP hdr	header	src port	dst port	seqno	ackno	flags	payload

- The dst port specifies which service will process the packet at the destination
  - Replies are sent back to the src port
- TCP is reliable, ordered, guaranteed (the TCP protocol looks after this for you)
- You can assume the packets you send are not lost and are received in order
- Before you can start, you must exchange 3 initiation packets

The HTTP Protocol

- HTTP is the protocol for interacting with web servers

ether hdr	IP hdr	TCP hdr	Request or reply

- Note that unlike the other protocols we have looked at, HTTP is a text-based protocol
  - “GET /index.html HTTP/1.0”

Putting it all together

- Open www.google.com in a browser
- We need to turn www.google.com into an IP address
  - We need to connect to our local domain server, 192.168.13.6
    - We need to get the MAC address for this address:
      - **Broadcast an ARP request** for 192.168.13.6's MAC address
      - **Receive ARP reply**
  - **Send a DNS request** to the DNS server's MAC address for www.google.com
  - **Receive DNS reply** saying it is at 203.8.170.1
- This is outside our network, so it must go via our gateway, 192.168.13.1
  - **Broadcast an ARP request** for 192.168.13.1's MAC address
  - **Receive ARP reply**
- Establish 3-way TCP handshake with 203.8.170.1 via our gateway MAC address
  - **Send TCP SYN** to port 80 on 203.8.170.1
  - **Receive TCP** SYN+ACK
  - **Send TCP ACK** to port 80 on 203.8.170.1
- **Send HTTP request** to port 80 on to 203.8.170.1: "GET /index.html HTTP/1.0"
- **Receive HTTP reply** containing HTML data
- Etc.

Summary

- Packet-based protocols are very flexible
  - TCP/IP has been around for over 40 years
  - There are hundredsof protocols based on it
- Once you understand the basic protocols we have covered, the rest is just details
- If you want to know more:
  - Anything by Richard Stevens is very readable
  - RFCs are Internet standards that define protocols exactly