**Introduction**

- Protocols are the rules for machine to machine communication.
- Well written protocols allow clients and servers written by different developers to communicate.
- Often, a protocol is specified in two major parts:
  - The structure of the message(s)
  - The sequencing of the messages
- In reality, a message between two processes needs to be compliant with a number of “layered” protocols. Protocols are layered to:
  - Simplify them
  - Allow one to be changed without impacting another

**Overview**

- OSI defines protocols at 7 levels:
  - 1 – Physical, 2 – Link
  - 3 – Network
  - 4 – Transport
  - 5 – Session, 6 – Presentation, 7 – Application
- The so-called local area network standards address layers 1, 2, and to some extent 3. This are 802.3, 802.4, 802.5, 802.7, 802.12b, etc.
- The Internet protocol (IP) is a classic level 3 protocol
- TCP is a classic level 4 protocol
- HTTP, SMTP, FTP, SNMP are all application protocols that handle/ don’t handle layers 5 and 6.
Protocol Common Sense (1)

• There are a number of possible hardware configurations -- for our purposes, consider everything is on an Ethernet bus.
• The connected machines all have an address and operate on a CSMA/CD basis – that is:
  • Each machine has a hardware enforced address – called a medium access control (MAC) address
  • The network connections use carrier sense multiple access (CSMA)
  • There is also a provision for collision detection (CD)

Protocol Common Sense (2)

• Using the MAC address, messages can be sent to a particular machine.
• If each machine is also given a logical address or name, we can now add a couple protocols:
  • A protocol can be added which identifies the source and destination logical address (for us this will be IP)
  • A protocol can be added which “resolves” or maps the logical address to the MAC address (for us this will be the ARP – Address Resolution Protocol)
  • A protocol can be added which “resolves” or maps the existence of multiple local area networks (for us this will be RIP – Routing Information Protocol)

Protocol Common Sense (3)

• We can overlay a high level protocol to add names to machines and translate them to IP addresses (for us DNS)
• We can add a protocol to operate on sets of packets, guaranteeing ordered and reliable delivery of multi-packet messages (for us TCP)
**Infrastructure Protocol Review**

- As always, reality is more complex, but for our purposes, we look at four frames that will carry our messages.
- Ethernet packets are the biggest envelop and carry our messages from point to point on the physical media.
- IP packets are the addressing packets over the large logical network. They exist as Ethernet packet data.
- TCP packets are the data in IP packets and they contain additional data for message structuring.
- Your application protocols exist as TCP data and they do the work you intend.

---

**Ethernet Packet**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble</td>
<td>64 bits</td>
</tr>
<tr>
<td>Source Address</td>
<td>48 bits</td>
</tr>
<tr>
<td>Destination Address</td>
<td>48 bits</td>
</tr>
<tr>
<td>Packet Length</td>
<td>16 bits</td>
</tr>
<tr>
<td>Message Data</td>
<td>368-12K</td>
</tr>
<tr>
<td>CRC</td>
<td>32 bits</td>
</tr>
</tbody>
</table>

- Preamble is a sequence of bits used to sync clocks
- Ethernet addresses are MAC addresses.
- MAC addresses are bound to network interface hardware.
- CRC is a data error checking code.
- The message is normally an IP packet.

---

**IP Packet**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver, IHL, Type of Service</td>
<td>Total Length (in octets)</td>
</tr>
<tr>
<td>Identifier, Flags</td>
<td>Fragment Offset</td>
</tr>
<tr>
<td>Time to Live, Protocol</td>
<td>Header Checksum</td>
</tr>
<tr>
<td>Source Address</td>
<td>Destination Address</td>
</tr>
<tr>
<td>Options and Padding</td>
<td>(to meet IHL spec)</td>
</tr>
<tr>
<td>Data/payload</td>
<td></td>
</tr>
</tbody>
</table>
**IP Packet Notes**

- **Ver** – IP Version(1-4)
- **IHL** – IP Header Length (# 32 bit words)
- **Type Service**
  - bits (0-2) network control, Inet control, Flash, Override; bit-3 = delay (norm, high); bit-4=throughput (norm, high);bit-5 = reliability(norm,high);bit-6/7 = reserved
- **Identifier, Flags, Offset** – used to reconstruct fragments
- **TTL** – normally max number of hops
- **Protocol** – UDP=17, TCP =6, etc.

---

**UDP Packet**

<table>
<thead>
<tr>
<th>1</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
<tr>
<td>Length</td>
<td>Checksum</td>
</tr>
<tr>
<td>Data</td>
<td></td>
</tr>
</tbody>
</table>

---

**TCP Packet**

<table>
<thead>
<tr>
<th>1</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
</tr>
<tr>
<td>Sequence Number (32)</td>
<td>ACK Number (32)</td>
</tr>
<tr>
<td>Offset</td>
<td>Reserved</td>
</tr>
<tr>
<td>Window</td>
<td>Checksum</td>
</tr>
<tr>
<td>Options and padding</td>
<td>Data (n)</td>
</tr>
</tbody>
</table>

Sequence Number for each octet sent

Flags:
- **U** - Urgent Pointer is significant
- **A** - ACK number field is significant
- **P** - Push Function (send all queued data)
- **R** - Reset the connection
- **S** - SYN - synchronize sequence numbers. When SYN Flag set = Initial Sequence Number (ISN)
- **F** - FIN - No more data from sender

Window - Number of octets receiver will accept before an ACK

Checksum - On parts of the header (pseudo-header)
TCP Setup Handshake

Client

Send SYN, seq=x

Receive SYN

Receive SYN+ACK

Send ACK y+1

Receive ACK

Server

Normal TCP Window

Client time Server

Send data 1-1000

Send data 1001-2000

Send data 2001-2500

ACK for 1000

ACK for 2000

ACK for 2500

Send data 2501-3500

Advertise window=2500

ACK to 1000, window 1500

ACK to 2000, window 500

ACK to 2500, window 0

Some Data Processed

ACK to 2500, window 2000

ACK to 3500, window 1000

Loss of ACK Timeout

Client

Send data 1-1000

Send data 1001-2000

Send data 2001-2500

ACK for 1000

ACK for 2000

ACK for 2500

Send data 2501-3500

ACK to 3500, window 1000

Server

X

Seq=92, 8 bytes data

ACK=100

ACK for 2500

Loss

Timeout

ACK=100
Late Response Timeout

Client
Seq=100, 20 bytes data
ACK=100
Seq=92 timeout

time

Server
Seq=92, 8 bytes data
ACK=120
Seq=92, 8 bytes data
Seq=100 timeout
ACK=120

Closing a connection

client
FIN

server
ACK
ACK
FIN

close

closed

closed

timed wait