Introduction to TCP/IP

Noah Mendelsohn
Tufts University
Email: noah@cs.tufts.edu
Web: http://www.cs.tufts.edu/~noah
What you should get from today’s session

- A high level introduction to TCP/IP and DNS
- By the end of this session you should have a basic understanding of:
  - IP Packets
  - IP Addresses
  - TCP Streams vs. UDP Datagrams
  - DNS and Domain Names
  - TCP/UDP Port numbers
Introduction to TCP/IP
## Architecture of the Internet Protocols

<table>
<thead>
<tr>
<th>Layer</th>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Program</td>
<td>Use the network for some purpose</td>
<td>Firefox, Apache Server, Your program</td>
</tr>
<tr>
<td>Application Layer</td>
<td>Protocols with application-specific semantics</td>
<td>HTTP (Web)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SMTP (E-mail)</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>User-level connection &amp; datagram</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>Internet Layer</td>
<td>Unreliable, multi-hop packet delivery</td>
<td>IP Packet Routing</td>
</tr>
<tr>
<td>Link Layer</td>
<td>Send an IP Packet over Hardware</td>
<td>Ethernet, Wi-fi, Dial-up</td>
</tr>
</tbody>
</table>
## Architecture of the Internet Protocols

<table>
<thead>
<tr>
<th>Layer</th>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Program</td>
<td>Use the network for some purpose</td>
<td>Firefox, Apache Server, Your program</td>
</tr>
<tr>
<td>Application Layer</td>
<td>Protocols with application-specific semantics</td>
<td>HTTP (Web) SMTP (E-mail)</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>User-level connection &amp; datagram</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>Internet Layer</td>
<td>Unreliable, multi-hop packet delivery</td>
<td>IP Packet Routing</td>
</tr>
<tr>
<td>Link Layer</td>
<td>Send an IP Packet over Hardware</td>
<td>Ethernet, Wi-fi, Dial-up</td>
</tr>
</tbody>
</table>
Internet Protocol (IP)
Internet Protocol (IP)

- Fundamental abstraction: best effort delivery of a single packet
- ...to anywhere in the Internet!
- Hides physical network differences / boundaries
  - Packets route uniformly through Ethernet, Wifi, Internet backbone, etc.
- Packets are sent to an IP Address
  - IPV4 addresses are 32 bits
  - Usually written: 130.64.212.28 (4 bytes, decimal)
- Fragmentation & reassembly
  - 65K maximum packet – in practice usually much smaller
  - Fragmentation supported by the protocol – inefficient and usually avoided
  - In practice: optimized systems use MTU discovery to send no more than what the path in question can handle without fragmentation (presumes stable paths!)
- Protocol field used to identify TCP vs. UDP, etc.
- Header is validity checked – content is not!
An IP V4 Packet

<table>
<thead>
<tr>
<th>V</th>
<th>HDLN</th>
<th>SVC TYPE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>FLGS</td>
<td>FRAG OFFSET</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>PROTOCOL</td>
<td>HDR CHECKSUM</td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESTINATION ADDRESS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THE TCP OR UDP DATA (VARIABLE LEN)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An IP V4 Packet

<table>
<thead>
<tr>
<th>V</th>
<th>HDLN</th>
<th>SVC TYPE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>FLGS</td>
<td>FRAG OFFSET</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>PROTOCOL</td>
<td>HDR CHECKSUM</td>
<td></td>
</tr>
</tbody>
</table>

- **V**: Version (4)
- **HDLN**: Header Length
- **SVC TYPE**: Service Type
- **LENGTH**: Total Length
- **ID**: Identification
- **FLGS**: Flags
- **FRAG OFFSET**: Fragment Offset
- **TTL**: Time to Live
- **PROTOCOL**: Protocol Type
- **HDR CHECKSUM**: Header Checksum
- **SOURCE ADDRESS**: Source Address
- **DESTINATION ADDRESS**: Destination Address
- **OPTIONS**: Options
- **THE TCP OR UDP DATA (VARIABLE LEN)**: Data

Packet Length (up to 65K “bytes”)
An IP V4 Packet

- **V**: Version number (4 bits)
- **HDLN**: Header length (4 bits)
- **SVC TYPE**: Service type (8 bits)
- **LENGTH**: Total length (16 bits)
- **ID**: Identification (16 bits)
- **FLGS**: Flags (16 bits)
- **FRAG**: Fragment offset (16 bits)
- **TTL**: Time to live (8 bits)
- **PROTOCOL**: Protocol type (8 bits)
- **HDR CHECKSUM**: Header checksum (16 bits)
- **SOURCE ADDRESS**: IP address of sender (32 bits)
- **DESTINATION ADDRESS**: IP address of receiver (32 bits)
- **OPTIONS**: Options (variable length)
- **THE TCP OR UDP DATA**: Variable length data for TCP or UDP (variable length)
An IP V4 Packet

Packet Fragmentation and Reassembly

<table>
<thead>
<tr>
<th>V</th>
<th>HDLN</th>
<th>SVC TYPE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>FLGS</td>
<td>FRAG OFFSET</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>PROTOCOL</td>
<td>HDR CHECKSUM</td>
<td></td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESTINATION ADDRESS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPTIONS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THE TCP OR UDP DATA (VARIABLE LEN)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## An IP V4 Packet

<table>
<thead>
<tr>
<th></th>
<th>32 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>HDLN</td>
</tr>
<tr>
<td>SVC</td>
<td>TYPE</td>
</tr>
<tr>
<td>TYPE</td>
<td>ID</td>
</tr>
<tr>
<td>FLGS</td>
<td>FRAG</td>
</tr>
<tr>
<td>OFFSET</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>PROTOCOL</td>
</tr>
<tr>
<td>SOURCE ADDRESS</td>
<td></td>
</tr>
<tr>
<td>DESTINATION ADDRESS</td>
<td></td>
</tr>
<tr>
<td>OPTIONS</td>
<td></td>
</tr>
<tr>
<td>THE TCP OR UDP DATA (VARIABLE LEN)</td>
<td></td>
</tr>
</tbody>
</table>

- **TCP? UDP?**
- **Note:** there is only space to name 256 choices
An IP V4 Packet

<table>
<thead>
<tr>
<th>V</th>
<th>HDLN</th>
<th>SVC TYPE</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>FLGS</td>
<td>FRAG OFFSET</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>PROTOCOL</td>
<td>HDR CHECKSUM</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>SOURCE ADDRESS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DESTINATION ADDRESS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>OPTIONS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>THE TCP OR UDP DATA (VARIABLE LEN)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Checksum guards header, not user data
Review: Internet Protocol (IP)

- **Fundamental abstraction**: best effort delivery of a single packet
- ...to anywhere in the Internet!
- **Hides physical network differences / boundaries**
  - Packets route uniformly through Ethernet, Wifi, Internet backbone, etc.
- **Packets are sent to an IP Address**
  - IPV4 addresses are 32 bits
  - Usually written: 130.64.212.28 (4 bytes, decimal)
- **Fragmentation & reassembly**
  - 65K maximum packet – in practice usually much smaller
  - Fragmentation supported by the protocol – inefficient and usually avoided
  - In practice: optimized systems use MTU discovery to send no more than what the path in question can handle without fragmentation (presumes stable paths!)
- **Protocol field used to identify TCP vs. UDP, etc.**
- **Header is validity checked** – content is not!
What about IP V6?

- **Same concept and unchanged TCP & UDP but…**

- **Much larger addresses: 128 bits**
  - Can in principle address $2^{128}$ items
  - The volume of the earth is approximately $2^{103}$ cubic millimeters!*
  - There are approximately $2^{81}$ stars in the known universe
  - You can likely address every bit of computer memory we would ever build, and pretty much every physical object of interest anywhere

- **Some other new options**
  - Network layer security
  - Optional *jumbograms* -- large packets for high speed links

- **New DNS “AAAA” records allow hostname $\rightarrow$ IP V6 address mapping**

* Statistics from http://www.wolframalpha.com
Introduction to the Domain Name System (DNS)
The Domain Name System

- Hosts can be given names like www.tufts.edu
- A standardized Internet service called the Domain Name System (DNS) provides a means of getting information about a DNS name
- In particular, DNS can get you the IP address(es) for a host name
- When you access a system or Web page based a name like www.tufts.edu, the DNS is almost surely being used first to find the IP address
- DNS is itself a UDP service*
- DNS can store other information, e.g. how to deal with email for a host, etc.

* Actually, for large requests TCP is used
The Domain Name System

- Invented by Paul Mockapetris in 1983
- Most important use: map domain names like cs.tufts.edu to IP addresses
- **DNS Names used as part of URIs** ([http://www.tufts.edu/index.html](http://www.tufts.edu/index.html)) and e-mail addr (noah@cs.tufts.edu)
- Actually DNS can store lots of information about a domain name
  - One or more IPV4 addresses (A records)
  - One or more IPV6 addresses (AAAA records)
  - Mail servers (MX)
  - Secure DNS (DNSKEY)
  - Etc
- Hierarchical resolution
DNS Resolution is Hierarchical

Not shown: your local machine will typically cache lookup results.

Look up .org

Look up wikipedia.org

Look up www.wikipedia.org (probably done at wikipedia)

…but how do we get started?

Not shown: your local machine will typically cache lookup results.

(public domain)
IDNA: Internationalized Domain Names

- Domain names are restricted to ASCII
- “On the wire” ASCII is used
- But…how to handle languages like Chinese?
- Kludge answer: Internationalized Domain Names (IDNA)
- Unicode characters are mapped using PunyCode to ASCII for use where real Domain Names are required
  - Example: Bücher.ch → xn--bcher-kva.ch
- Browsers, etc. recognize the IDNA forms and present Unicode
- First non-ASCII top level domains registered in 2009
Summary: resolution & registration of domain names

- Your machine probably has a local resolver that caches DNS lookups.
- You also usually configure your machine with the address of a DNS server that can help look up new names.
- Caching is done at every level, but a full resolution starts by going to a so-called “root” server, which knows servers for common domains like “.com”, “.edu”, etc.
- The DNS server for “.edu” has an entry showing the IP address(es) of DNS servers maintained by Tufts.
- Within the Tufts DNS server, there is an entry for “www”, and it has the IP address to which requests for Web pages like `http://www.tufts.edu` should be sent.

*Note that registration is delegated: registering a new Top Level Domain (.com) is a big deal; adding linux.eecs to Tufts.edu can be handled locally at Tufts.*
## Architecture of the Internet Protocols

<table>
<thead>
<tr>
<th>Layer</th>
<th>Purpose</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Program</td>
<td>Use the network for some purpose</td>
<td>Firefox, Apache Server, Your program</td>
</tr>
<tr>
<td>Application Layer</td>
<td>Protocols with application-specific semantics</td>
<td>HTTP (Web) SMTP (E-mail)</td>
</tr>
<tr>
<td>Transport Layer</td>
<td>User-level connection &amp; datagram</td>
<td>TCP/UDP</td>
</tr>
<tr>
<td>Internet Layer</td>
<td>Unreliable, multi-hop packet delivery</td>
<td>IP Packet Routing</td>
</tr>
<tr>
<td>Link Layer</td>
<td>Send an IP Packet over Hardware</td>
<td>Ethernet, Wi-fi, Dial-up</td>
</tr>
</tbody>
</table>
User-level Protocols
Two common choices for the transport protocol

- UDP – use-level unreliable packets
- TCP – user-level reliable, flow-controlled streams

Both provide connectivity between applications anywhere on the Internet
User Datagram Protocol (UDP)
User Datagram Protocol - UDP

- Lets **user programs** send/receive unreliable *datagram* messages
- Messages may be dropped or arrive out of order
- Length is preserved – message boundaries maintained
- …isn’t that the same as IP? No!
- UDP is *program-to-program*, not host-to-host!
- Delivery is unreliable, but content is checksummed: if it arrives, it’s clean
- Length limited only by IP (but usually applications set a 512byte max)
- UDP was designed by David Reed (the same one who wrote the “End-to-end” paper)

* [http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml](http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml)
Addressing TCP & UDP Communications

- The addresses in the IP packet only identify a host computer.
- The protocol field picks TCP vs UDP, etc.
- But... which of the many possible receivers at that host are you talking to?
- Answer: each TCP or UDP packet is addressed to an IP-Address:port pair.
- The port is in the TCP or UDP part of the IP packet.
- Well known ports identify common servers like e-mail (587) and Web (80).
- Dynamically allocated ports are used, e.g., for response traffic.
- Setting up ports for test purposes is a mess...see COMP 150-IDS instructions for getting ports to use for testing your programs.

Port 80 is famous in the Web world – this is a detail you’ll be expected to remember!
Advantages and disadvantages of UDP

- **Advantages**
  - Leverages end-to-end: applications can tune protocols for specific needs
  - No setup overhead: typically 1 message $\rightarrow$ 1 IP packet
  - Very efficient for small, *idempotent* messages
Brief interruption to explain *idempotence*

- **Advantages**
  - Leverages end-to-end: applications can tune protocols for specific needs
  - No setup overhead: typically 1 message → 1 IP packet
  - Very efficient for small, *idempotent* messages

*Idempotence*

*Crucial concept in system & protocol design.*

An *idempotent* operation yields the same result no matter how many times it’s executed. *Example: retrieve a value*

*Idempotent operations can be retried without harm*
Advantages and disadvantages of UDP

- **Advantages**
  - Leverages end-to-end: applications can tune protocols for specific needs
  - No setup overhead: typically 1 message → 1 IP packet
  - Very efficient for small, *idempotent* messages
  - Example: look up a DNS record*

- **Disadvantages**
  - Inventing a non-trivial custom protocol over UDP is almost always a mistake
  - Getting things like flow control and setup/teardown right is tricky – TCP does it for you
  - TCP provides a reliable, well-tuned universal implementation of reliable streams over IP
Transmission Control Protocol (TCP)
TCP

- The standard way of sending reliable streams of data over the Internet
- *The basis for most Internet application protocols including HTTP*
- Same port-addressing architecture as UDP
- Protocols carefully tuned over many years to handle
  - Wide variety of network speeds, MTUs etc.
  - Retry, congestion control etc.
- **Message boundaries not preserved: just bidirectional byte streams**
- On Unix & Linux: read/write APIs compatible with file & pipe read/write
  - In some cases, code need not know whether it’s using a file or a network socket
TCP Checks and Sequences Packets to create Streams

TCP creates reliable, end-to-end streams from unreliable IP packets (datagrams)
TCP/IP: Review Summary

- Each node (machine) is given a 4 byte “IP Address” – e.g. 130.64.23.39
- DNS provides symbolic names for hosts (E.g. linux.eecs.tufts.edu)
- IP layer provides unreliable, unordered delivery of packets
  - Packets can be up to 65K bytes, but usually smaller
  - Note that each packet has source/destination IP address, checksum to protect the header (not the data!), and a length field
- TCP provides reliable, ordered streams of unlimited length
  - TCP streams are used by most Internet applications, including the Web
  - Built on top of IP: TCP provides the necessary connection setup, sequencing, timeout/retry, data integrity checks, etc.
- UDP provides for addressing and delivery of unreliable, unordered datagrams for applications (IP is typically host-to-host)
- The senders and receivers of TCP & UDP traffic are identified by (IP Address, port), where port is a 16 bit number (Web servers conventionally respond on port 80, SMTP mail uses port 25, etc.)

For simplicity, the above describes the older and more widely deployed IPV4. IPV6 enables much larger addresses, and many other features.