Chapter 7.B and 7.C

Link Layer & Network Layer

Prof. Dina Katabi

Some slides are from lectures by Nick McKeown, Ion Stoica, Frans Kaashoek, Hari Balakrishnan, and Sam Madden
Previous Lecture

We learned how to share the network infrastructure between many connections/flows

We also learned about the implications of the sharing scheme (circuit or packet switching) on the service that the traffic receives
This Lecture

An Example: HTTP
Layering
Link Layer
Example: HTTP over the Internet Using TCP/IP and Ethernet
In the sending host

1. **Application-Programming Interface (API)**
   Application requests TCP connection with “B”

2. **Transmission Control Protocol (TCP)**
   
   ![Diagram of TCP packet structure]
   
   - TCP Packet
   - TCP requests IP packet to be sent to “B”
   - Empty in Connection Setup Packet
   - Type = Connection Setup
In the sending host (2)

3. Internet Protocol (IP)

Creates IP packet with correct addresses. IP requests packet to be sent to router.

```
In the sending host (2)

3. Internet Protocol (IP)

Creates IP packet with correct addresses. IP requests packet to be sent to router.

TCP Packet

<table>
<thead>
<tr>
<th>TCP Data</th>
<th>TCP Header</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Encapsulation

<table>
<thead>
<tr>
<th>IP Data</th>
<th>IP Header</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Destination Address: IP “B”
Source Address: IP “A”
Protocol = TCP

IP Packet
```
In the sending host (3)

4. Link ("MAC" or Ethernet) Protocol

Creates MAC frame.
Wait for access to the line.
Send each bit of the frame.
**In Router R1**

5. **Link ("MAC" or Ethernet) Protocol**

Accept MAC frame, check address and Frame Check Sequence (FCS) to ensure no bit errors. Pass data to IP Protocol.

![Diagram of IP and Ethernet packets with Decapsulations and address details](image)

- **IP Packet**
  - IP Header
  - IP Data

- **Ethernet Packet**
  - Ethernet Header
  - Ethernet Data
  - FCS

- **Decapsulation**

  - Destination Address: MAC “R1”
  - Source Address: MAC “A”
  - Protocol = IP
6. *Internet Protocol (IP)*

Use IP destination address to decide where to send packet next ("next-hop routing"). Request Link Protocol to transmit packet.

```
IP Packet
```

```
<table>
<thead>
<tr>
<th>IP Data</th>
<th>IP Header</th>
</tr>
</thead>
</table>

Destination Address: IP “B”
Source Address: IP “A”
Protocol = TCP
In Router R1

7. Link ("MAC" or Ethernet) Protocol

Creates MAC frame.
Wait for access to the line.
Send each bit of the frame.

IP Packet

<table>
<thead>
<tr>
<th>IP Data</th>
<th>IP Header</th>
</tr>
</thead>
</table>

Encapsulation

| Ethernet FCS | Ethernet Data | Ethernet Header |

Destination Address: MAC "R2"
Source Address: MAC "R1"
Protocol = IP
Steps 8-15 are the same as before ...

In Router R5

16. Link ("MAC” or Ethernet) Protocol

- Creates MAC frame.
- Wait for access to the line.
- Send each bit of the frame.

IP Packet

<table>
<thead>
<tr>
<th>IP Data</th>
<th>IP Header</th>
</tr>
</thead>
</table>

Encapsulation

<table>
<thead>
<tr>
<th>Ethernet FCS</th>
<th>Ethernet Data</th>
<th>Ethernet Header</th>
</tr>
</thead>
</table>

Destination Address: MAC “B”
Source Address: MAC “R5”
Protocol = IP
In the receiving host

17. Link ("MAC" or Ethernet) Protocol
Accept MAC frame, check address and Frame Check Sequence (FCS) for bit errors.
Pass data to IP Protocol.

<table>
<thead>
<tr>
<th>Ethernet FCS</th>
<th>Ethernet Data</th>
<th>Ethernet Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decapsulation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Destination Address: **MAC “B”**
Source Address: **MAC “R5”**
Protocol = IP
In the receiving host (2)

18. Internet Protocol (IP)

Verify IP address.
Extract/decapsulate TCP packet from IP packet.
Pass TCP packet to TCP Protocol.

TCP Packet

Decapsulation

IP Packet

<table>
<thead>
<tr>
<th>TCP Data</th>
<th>TCP Header</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>IP Data</th>
<th>IP Header</th>
</tr>
</thead>
</table>

Destination Address: IP “B”
Source Address: IP “A”
Protocol = TCP
In the receiving host (3)

19. **Transmission Control Protocol (TCP)**
   Accepts TCP “Connection setup” packet
   Establishes connection by sending “Ack”.

20. **Application-Programming Interface (API)**
   Application receives request for TCP connection with “A”.

---

![TCP Packet Diagram]

- **TCP Packet**
  - **TCP Data**
  - **TCP Header**
  - Type = Connection Setup
  - Empty
This Lecture

An Example: HTTP
Layering
Link Layer
Network Architecture

Problem

Networks are complex (heterogeneity, distributed, delay, losses, reordering, ...)

How do we organize a network implementation?

Solution

To deal with complexity use layering
Layering

Layering is a particular form of abstraction

The system is broken into a vertical hierarchy of logically distinct entities (layers)

The service provided by one layer is based solely on the service provided by layer below
Layering: Our HTTP Example

The 4-layer Internet model

HTTP
TCP
IP
Ethernet

Application
Transport
Network
Link

End-to-End Layer
Who Does What?

Link Layer:
Delivers data from one end of a link to the other

Network Layer
Routes packets and delivers them to their destination

Transport Layer
Provides useful abstractions: stream, message
Can provide reliability
Can provide congestion control

Application Layer
Your application: HTTP, FTP, etc.
Where are these layers?

Link and network layers are implemented everywhere.

The end-to-end layer (i.e., transport and application) is implemented only at hosts.
Encapsulation

A layer can use only the service provided by the layer immediate below it
Each layer may change and add a header to data packet
Interface

Higher layer calls lower layer
  e.g., Link_Send(this_data, this_link)

Lower layer uses an up-call function to inform the higher layer of data arrival
  e.g., Network_Handle()
Multiplexing in the Internet

Many possible applications, transports, and link layers

But they all use IP at the network layer
This Lecture

An Example: HTTP
Layering
Link Layer
**Problem:**
Deliver data from one end of the link to the other

**Need to address:**
Bits Analog Bits
Framing
Errors
Medium Access Control (The Ethernet Paper)
Sending bits

Bits      Analog      Bits
Receiver needs to detect the value of the bits
Manchester Encoding: each bit is a transition
                      Having a transition in each bit allows the receiver to synchronize to the sender’s clock
Framing

Receiver needs to detect the beginning and the end of a frame

Use special bit-pattern to separate frames
  E.g., pattern could be 1111111 (7 ones)

Bit stuffing is used to ensure that a special pattern does not occur in the data
  If pattern is 1111111 Whenever the sender sees a sequence of 6 ones in the data, it inserts a zero (reverse this operation at receiver)
Error Handling

Detection:

Use error detection codes, which add some redundancy to allow detecting errors

When errors are detected

Correction:

• Some codes allow for correction

Retransmission:

• Can have the link layer retransmit the frame (rare)

Discard:

• Most link layers just discard the frame and rely on higher layers to retransmit
This Lecture

To cope with the complexity, the network architecture is organized into layers

The link layer delivers data between two machines that are directly connected using a link