**CS 455: Introduction to Distributed Systems**

**[Networking]**

**The Receiver’s Buffer**

Small it may be
But throttle the mightiest sender it can
Not just the how much
But also the when
Or if at all

Shrideep Pallickara
Computer Science
Colorado State University

---

**Frequently asked questions from the previous class surveys**

- When are trailers necessary?
- Are encryptions only on the payloads?
- What is connectionless?
- TCP vs UDP

---

**Topics covered in today’s lecture**

- IP routing
- IPv6
- UDP
- TCP

---

**Every network type has a Maximum Transmission Unit (MTU)**

- Largest IP datagram that it can carry in its frame
- This is smaller than the largest packet-size of network
  - IP datagram needs to fit in the payload of the link-layer frame

---

**IPv4 Packet header**

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>Header</td>
<td>4</td>
</tr>
<tr>
<td>Data</td>
<td>20</td>
</tr>
</tbody>
</table>

---

**Fragmentation necessary when datagram path includes network with smaller MTU**

- All fragments carry same identifier in **Ident** field
- To enable fragment reassembly
- Chosen by the source host
- If all fragments do not arrive at receiving host?
  - Receiver gives up reassembly (reassembly timeout: 15 seconds RFC0791)
  - **Discards** fragments that did arrive
- IP does not attempt to recover from missing fragments
Header fields used in IP fragmentation:
Fragmentation occurs at 8-byte boundaries

IPv6 (AND COMPAREING WITH IPv4)

IPv6 Packet Header: Some more details [1/2]

IPv6 Packet Header: Some more details [2/2]
IPv6 Packet: Some more details [2/2]

- Next Header (8 bits)
  - Specifies the type of the next header
- Hop Limit (8 bits)
  - Replaces the time-to-live field of IPv4
- Destination and Source Addresses (128 bits or 16 bytes each)
  - Note: The IPv6 packet header has no checksum
- Transport or application layer protocols are assumed to provide sufficient error detection

Structure of the IPv6 Packet

PDU typically contains an upper layer protocol header and its payload.
For e.g.: a TCP segment, UDP Datagram, or an ICMPv6 message

IPv6 Extension Headers: The chain of pointers using the Next Header field

Each extension header must fall on a 64-bit (8-byte) boundary. Use Padding to get there if less than that.

User Datagram Protocol

- Simplest possible transport protocol
- Extends host-to-host into process-to-process communications
- No additional functionality to best-effort service provided by underlying network
- Adds demultiplexing
  - Allows applications on a host to share the service
UDP identification of processes

- Processes indirectly identify each other
  - Abstract locator called port
- Source sends a message to a port
  - Destination receives messages from a port
- Process is identified by a port on a particular host

Format of a UDP header

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>16</td>
</tr>
<tr>
<td>DstPort</td>
<td>16</td>
</tr>
<tr>
<td>Length</td>
<td>8</td>
</tr>
<tr>
<td>Checksum</td>
<td>16</td>
</tr>
<tr>
<td>Data</td>
<td>0</td>
</tr>
</tbody>
</table>

A port is just an abstraction

- Typically implemented as a message queue
- When message arrives?
  - Protocol appends message to end of the queue
- UDP
  - If the queue is full, message is discarded
  - No flow-control mechanism

Some work that UDP does do besides demultiplexing: Checksumming

- UDP header
- Message body
- Pseudohedger: From the IP header
  - Protocol number
  - Source IP address
  - Destination IP address
- UDP length
- Used twice

Some work that UDP does do besides demultiplexing: Checksumming

- Verify if message is delivered between the correct endpoints

UDP message queue: The port abstraction

Reliable Byte Stream

TCP
Components of Reliable delivery

- **Acknowledgements**
  - Confirm receipt of data

- **Timeouts**
  - Retransmit if ACK not received within a specified time

- **Use of ACKs and timeouts to implement reliable delivery**
  - Sometime called ARQ (automatic repeat request)

Simplest ARQ is the stop-and-wait algorithm

- After transmitting one frame
  - Sender waits for ACK before transmitting the next frame

- If the ACK does not arrive after a period of time
  - Sender retransmits the original frame

### Stop-and-wait [1/2]

- **Timeline for the sliding window**

### Sliding window: Try to fill the network pipe

- **Delay x Bandwidth product is 8 KB**

- Data frames = 1KB

- Sender could transmit 9th frame
  - When ACK for the 1st frame arrives

### Timeline for the sliding window
Sliding Window on Sender/Receiver

- Send Window Size
  - $\text{Last ACE Received}$ to $\text{Last Frame Sent}$
  - $\geq$ Send Window Size

- Receive Window Size
  - $\text{LAR}$ (Last ACK Received) to $\text{LAF}$ (Largest Acceptable Frame)
  - $\geq$ Receive Window Size

Transmission Control Protocol (TCP)

- Reliable, in-order delivery of byte streams
- Full duplex protocol
  - Each connection supports a pair of byte streams
  - Flowing in different directions
- Includes flow control mechanism
  - Allows receiver to limit the data sender
  - Control how much data can be transmitted at a time

TCP: Setup and Teardown

- Two sides of the connection agree to exchange data
  - Establish shared state
  - 3 packets exchanged (SYN, SYN-ACK, ACK)
- Connection teardown
  - Let each host know it is OK to free the shared state
  - 4 packets exchanged (FIN, ACK, FIN, ACK)

Transmission Control Protocol (TCP)

- Includes multiplexing mechanism
  - Multiple applications on a given host
- Implements a congestion-control mechanism
  - Throttle how fast TCP sends data
  - Keep sender from overloading the network

Flow control and congestion control

- Flow control is an end-to-end issue
  - Don’t overrun capacity of receiver
- Congestion control is about hosts & networks interact
  - Don’t cause switches and links to be overloaded

TCP Segments & how they come about

- TCP
  - Accepts data from a data stream
  - Breaks it up into chunks
  - Adds a TCP header $\ldots$ creating a TCP segment
  - Segment is then encapsulated in a IP datagram
- TCP packet is a term that you will often hear
  - Segment is more precise, packets are generally datagrams, frames are at the link layer
How TCP manages a byte stream

TCP Header Format

Relationship between SequenceNum, Acknowledgement and AdvertisedWindow

TCP Sliding Window

TCP Send Buffer
**TCP Receive Buffer**

- **LastByteRead** < **NextByteExpected**
- **Space Utilized in the receiver’s buffer**

**Flow Control: Buffers are of finite size**

MaxSendBuffer and MaxRcvBuffer

- Receiver throttles sender
  - Advertises a window
  - No bigger than what it can buffer

- LastByteRcvd - LastByteRead ≤ MaxRcvBuffer
- AdvertisedWindow = MaxRcvBuffer - (\(\text{NextByteExpected} - 1\) - LastByteRead)

**Flow Control: Buffers are of finite size**

MaxSendBuffer and MaxRcvBuffer

- On the sender side, TCP adheres to the advertised window from the receiver

- LastByteSent - LastByteAcked ≤ AdvertisedWindow

- EffectiveWindow = AdvertisedWindow - (LastByteSent - LastByteAcked)

- EffectiveWindow should be > 0 before source can send more data

**The advertised window may potentially shrink**

- If the process is reading data as fast as it arrives?
  - The advertised window stays open
  - i.e., AdvertisedWindow = MaxRcvBuffer

- If the receiving process falls behind?
  - Advertised window becomes smaller with every segment that arrives
  - Until it becomes 0

**The contents of this slide-set are based on the following references**

- Understanding the IPv6 Header: [Link](https://www.microsoftpressstore.com/articles/article.aspx?p=2225063&seqNum=4)