COMP90007 Internet Technologies Workshop

Week 8 – Transport Layer

Ruixi
huor@student.unimelb.edu.au

Review

Encapsulation

Service

The ultimate goal of the transport layer is **to provide efficient, reliable, and cost-effective data transmission service** to its users, normally processes in the application layer.

To achieve this, the transport layer makes use of the services provided by the network layer.

2 types of services

Transport layer adds *reliability* to the network layer

- Offers connectionless (e.g., UDP) and connection-oriented (e.g, TCP) services to applications

Transport Layer and Network Layer Services Compared

- If transport and network layers are so similar, why are there two layers?
- Transport layer code runs entirely on hosts
- Network layer code runs almost entirely on routers
- Transport layer can fix reliability problems caused by the Network layer (e.g., delayed, lost or duplicated packets)
- Users have no real control over the network layer – Transport layer: improve QoS

Connection-Oriented Service

Connection establishment -- Three Way Handshake

How sequence and acknowledgement numbers work

The TCP Segment Header

- TCP header includes addressing (ports), sliding window (seq. / ack. number), flow control (window), error control (checksum) and more

Connection-Oriented Service

Transport layer provides application developer abstraction/primitives to simplify the work

Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	Request a release of the connection

Figure 6-2. The primitives for a simple transport service.

Connection Release

- Asymmetric Disconnection
  - Either party can issue a DISCONNECT, which results in DISCONNECT TPDU and transmission ends in both directions
- Symmetric Disconnection
  - Both parties issue DISCONNECT, closing only *one direction at a time* -allows flexibility to remain in receive mode

- Asymmetric vs Symmetric connection release types
- Asymmetric release may result in data loss hence symmetric release is more attractive
- Symmetric release works well where each process has a set amount of data to transmit and knows when it has been sent

Question 1

Why does the maximum packet lifetime, *T*, have to be large enough to ensure that not only the packet but also its acknowledgements have vanished?

Answer:
Look at the second duplicate packet in Fig. 6-11(b). When the packet arrives, it would be a disaster if acknowledgements to *y* were still floating around.

Question 2

Imagine that a two-way handshake, rather than a three-way handshake were used to set up connections. In other words, the third message was not required. Are deadlocks now possible? Give an example or show that none exist.

Answer:

Deadlocks are possible.

For example, a packet arrives at A out of the blue, and A acknowledges it. The acknowledgement gets lost, but A is now open while B knows nothing at all about what has happened. Now the same thing happens to B, and both are open, but expecting different sequence numbers.

Timeouts have to be introduced to avoid the deadlocks.

Question 3

- Transport layer code runs entirely on hosts

Why does UDP exist? Would it not have been enough to just let the user processes send raw IP packets?

Answer:

No. IP packets contain **IP addresses, which specify a destination machine**.

Once such a packet arrived, how would the network handler know which process to give it to? **UDP packets contain a destination port**.

This information is essential so they can be delivered to the correct **process**.

Question 4

1 gigabit = 10^9 bits

A client sends a 128 byte request to a server located 100 km away over a 1 gigabit optical fibre. What is the efficiency of the line during the remote procedure call?

Assuming **propagation speed** in fibre is 2/3 the speed of light, 2×10^8 m/s

efficiency = transmission delay / (transmission delay + propagation delay)

Transmission delay is how long it takes to get all the bits into the wire in the first place (it's **packet_length/data_rate**).

- Sending 1024 bits over a 1 Gbps line takes about $1024/10^9 = 10^{-6}$s = 1 µs

- propagation speed = 2×10^8 m/s, then the **propagation delay** will be: $2\times100\times10^3$ m / 2×10^8 m/s = 1 ms

- Efficiency = 1 µs / 1 ms = 0.1%

Question 5

Both UDP and TCP use port numbers to identify the destination entity when delivering a message. Give two reasons for why these protocols invented a new abstract ID (port numbers), instead of using process IDs, which already existed when these protocols were designed?

Answer:

Here are three reasons.

- First, process IDs are **OS-specific**. Using process IDs would have made these protocols OS-dependent.
- Second, **a single process may establish multiple channels** of communications. A single process ID (per process) as the destination identifier cannot be used to distinguish between these channels.
- Third, having processes listen on **well known ports** is easy, but well-known process IDs are impossible.

Question 6

Datagram fragmentation and reassembly are handled by IP and are invisible to TCP. Does this mean the TCP does not have to worry about data arriving in the wrong order?

Answer:

Even though *each datagram* arrives intact, it is possible that the datagrams arrive in the wrong order, so TCP has to be prepared to reassemble the parts of the message properly.

Question 7

A process on host 1 has been assigned port *p*, and a process on host 2 has been assigned port *q*. Is it possible for there to be two or more TCP connections between these two ports at the same time?

Sender and receiver both create "sockets", consisting of the IP address of the host and a port number

Socket at host 1: **(1, p)**

Socket at host 2: **(2, q)**

A connection is identified only by its sockets.
Thus, **(1, p) → (2, q)** is the only possible connection between those two ports.

Question 8

The maximum payload of a TCP segment is 65,495 bytes. Why was such a strange number chosen?

Segment payload = frame payload – packet header – segment header

IP packet length
(includes header and payload)

IPv4 Frame Structure

16 bits

The maximum length = 1111111111111111 =65535

Question 8

The maximum payload of a TCP segment is 65,495 bytes. Why was such a strange number chosen?

Segment payload = frame payload – packet header – segment header

IP packet length
(includes header and payload)

65535 bytes

IPv4 Frame Structure

Minimum IP packet header = 5 * 4 = 20bytes

The TCP Segment Header

Minimum TCP segment header = 5 * 4 = 20bytes

- TCP header includes addressing (ports), sliding window (seq. / ack. number), flow control (window), error control (checksum) and more

Question 8

The maximum payload of a TCP segment is 65,495 bytes. Why was such a strange number chosen?

Segment payload = frame payload – packet header – segment header = 65495 bytes

20 bytes **20 bytes**

IP packet length
(includes header and payload)

65535 bytes

Question 9

1 Tbps = 10^{12}bps = $1.25*10^{11}$ bytes/second

To get around the problem of sequence numbers wrapping around while old packets still exist, one could use 64 bit sequence numbers. However, theoretically, an optical fibre can run at 75 Tbps.

What maximum packet lifetime is required to make sure that future 75 Tbps networks do not have wraparound problems even with 64 bit sequence numbers? Assume that each byte has its own sequence number, as TCP does.

Answer:

- The sequence space is 2^{64}. Each byte is assigned a unique sequence number, so the sequence numbers can represent 2^{64} unique bytes.
- A 75 Tbps transmitter can output 9.375×10^{12} bytes/s.
- $t = 2^{64} / 9.375 \times 10^{12} = 1.96 \times 10^{6}$ s = 22.77 days
- It takes 22.77 days, which is over three weeks, to wrap around. A maximum packet lifetime of less than 22.77 days will prevent the problem. In short, going to 64 bits is going to work for quite a while.