Computer Networks

Grado en Ingeniería Informática

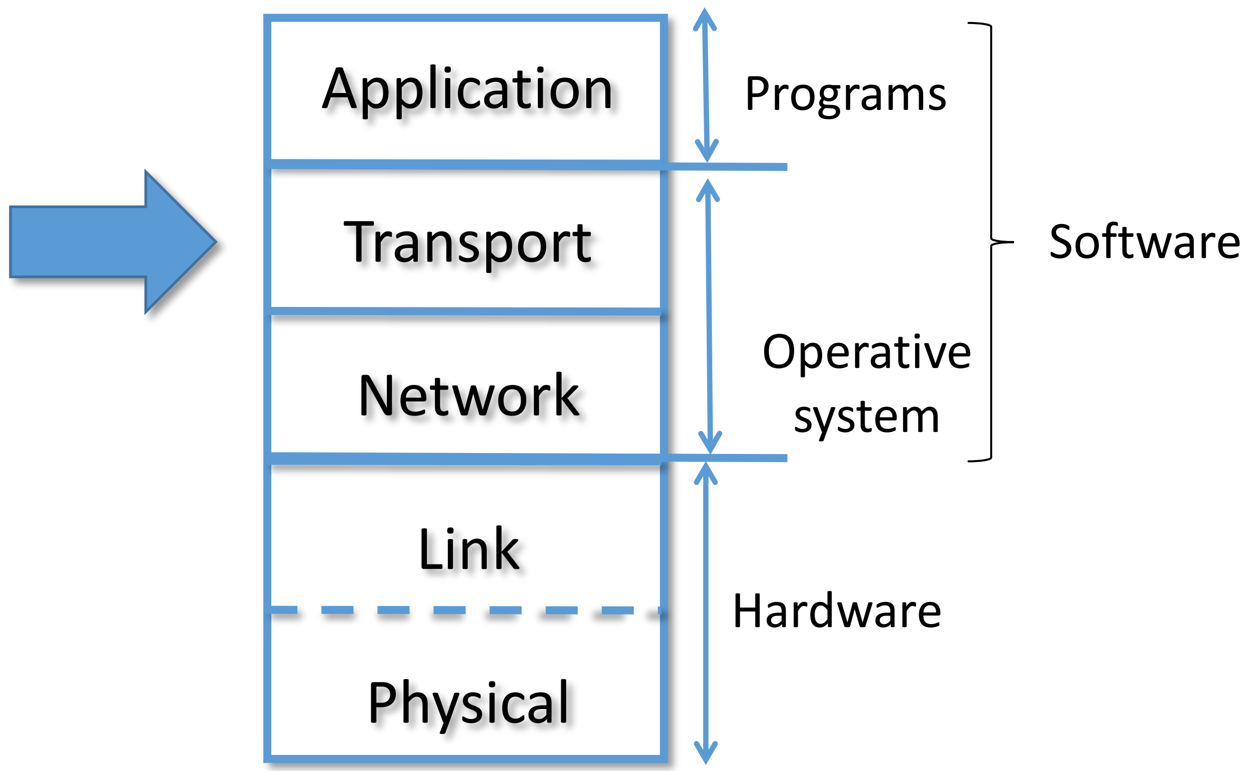
Universidad de Sevilla

Departamento de Tecnología Electrónica

Computer Networks
Lesson 3

The Transport Layer

Departamento de
Tecnología Electrónica


Application
Transport
Network
Link
Physical
Programs
Operative system
Hardware
Software

Lesson 3: The Transport Layer

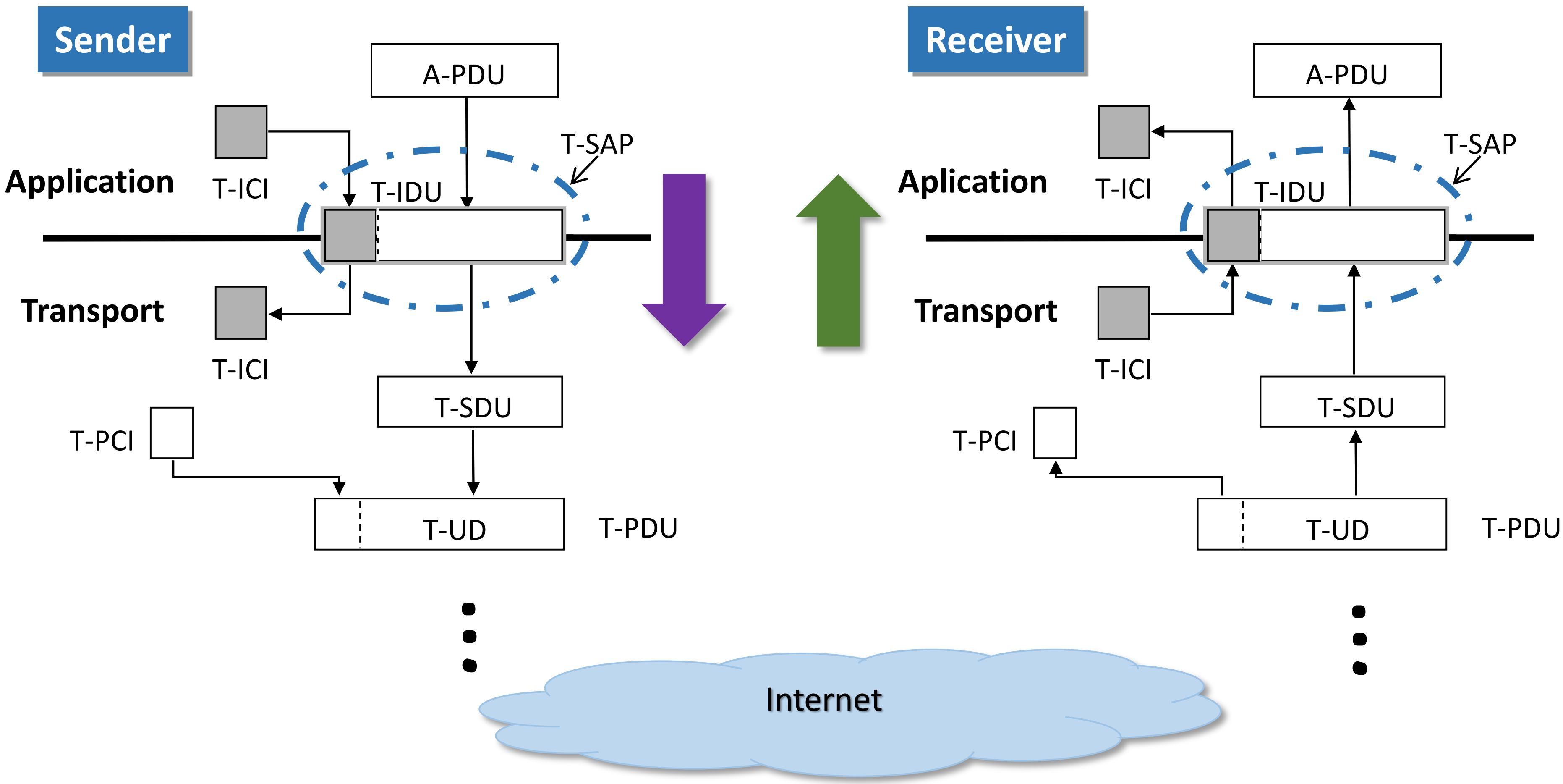
Objectives

- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

1. **Transportation Level Services**
2. Multiplexing and demultiplexing
3. No conexion transport: UDP
4. Principles of reliable transfer
5. Connection-oriented transport: TCP
   - TCP segment structure
   - Reliable data transfer
   - Flow control
   - Connection management

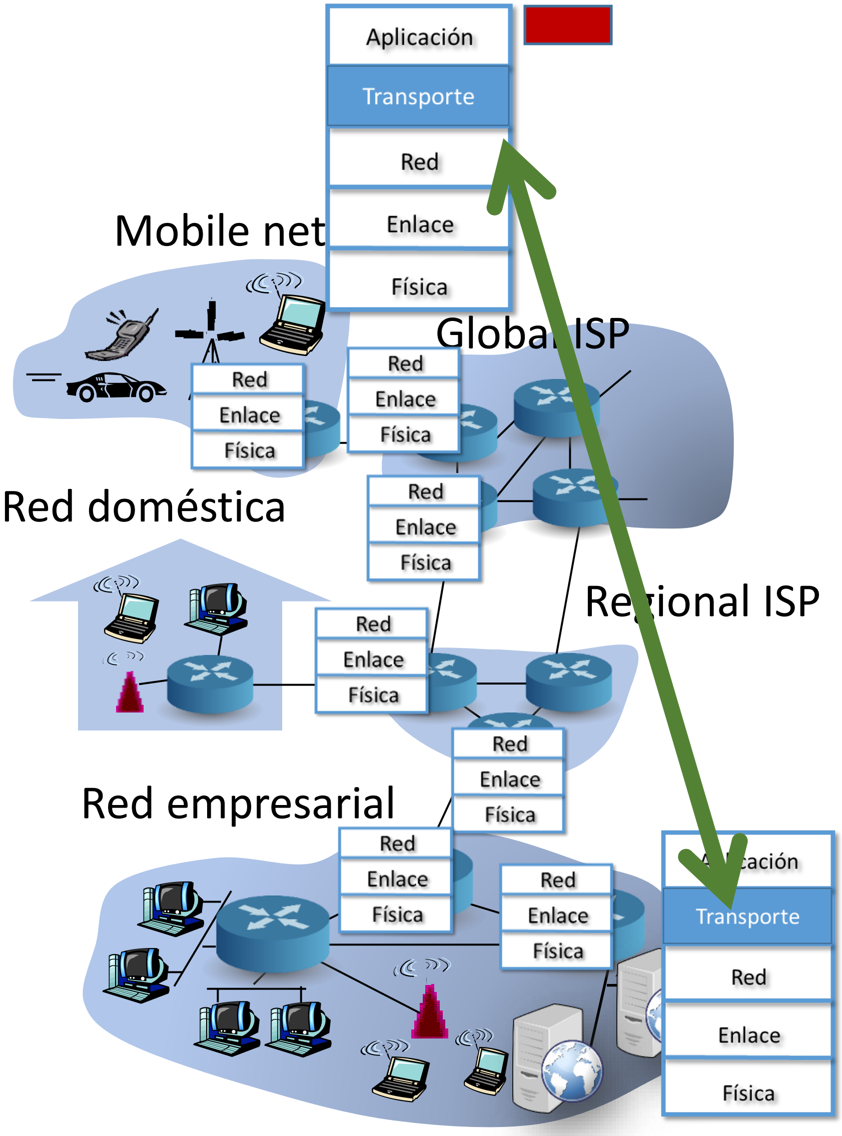
Transport Layer Services
Sockets



Transportat Layer Services

Internet Transport Protocols

- **TCP**: Reliable and in-order data delivery service
  - Flow control
  - Congestion control
  - Establishing the connection
- **UDP**: Unreliable data delivery service without order guarantee.
- Simple protocol
- Services not available:
  - Guaranteed delay
  - Guaranteed bandwidth
- both tcp and udp use ip services
- Protocol that provides a best-effort service



Transport Layer Services
Transport vs. Network

- **Network layer**: provides a logical communication service between end computers(hosts)
- It allows (thanks to IP addresses) to identify a final system on the Internet.
- It is a service similar to the postal service that allows you to send a letter to a house (address).
- **Transport layer**: extends the network layer service to provide a logical communication service between application processes.
- It allows different processes in the same end system to use the same network level thanks to the ports (this is known as transport layer multiplexing and demultiplexing).
  - It is achieved using the service provided by the network layer, to, from it, build an "improved" service.

Lesson 3: The Transport Layer

Objectives

- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

1. Transportation Level Services
2. **Multiplexing and demultiplexing**
3. No conexion transport: UDP
4. Principles of reliable transfer
5. Connection-oriented transport: TCP
   - TCP segment structure
   - Reliable data transfer
   - Flow control
   - Connection management

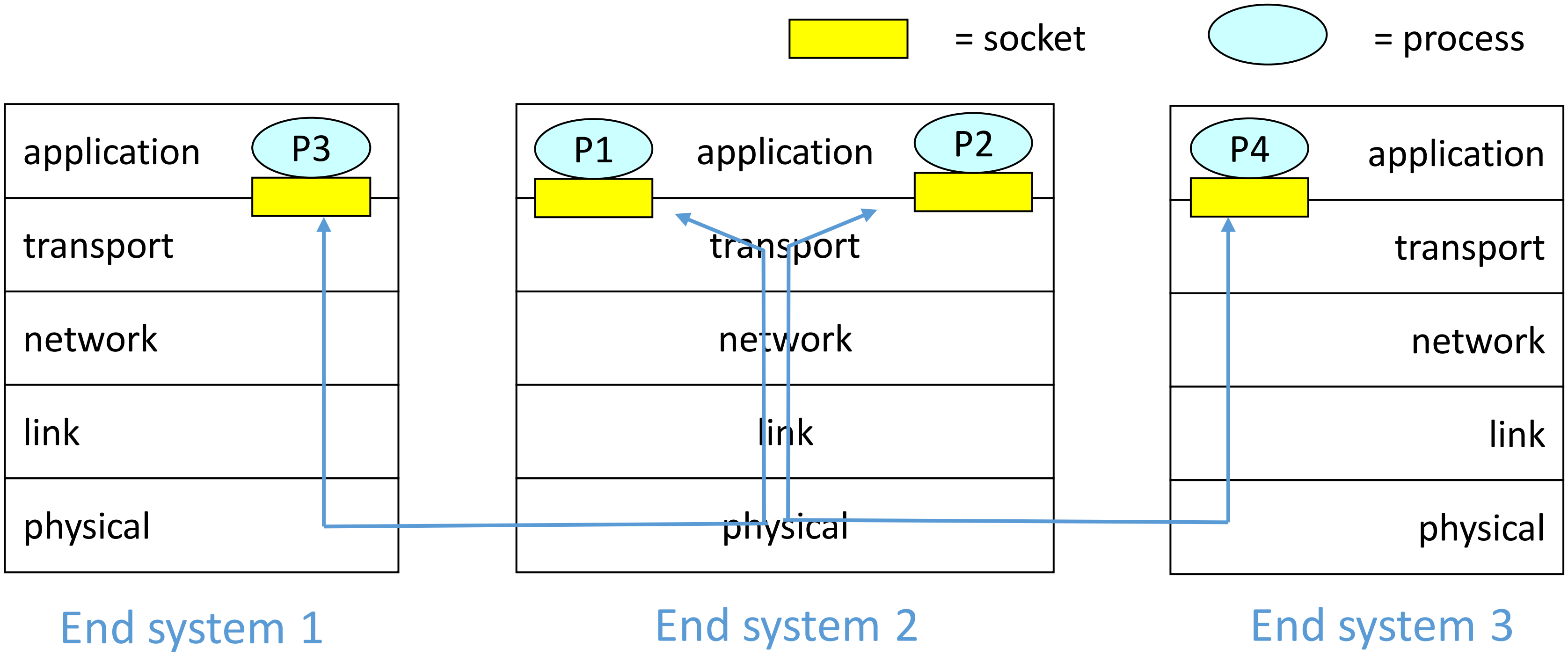
Multiplexing and Demultiplexion

Multiplexing when sending

Collect data from multiple sockets, create the segments (T_PDU) by adding header information (T_PCI) that will be used later when demultiplexing.

Demultiplexion upon receipt

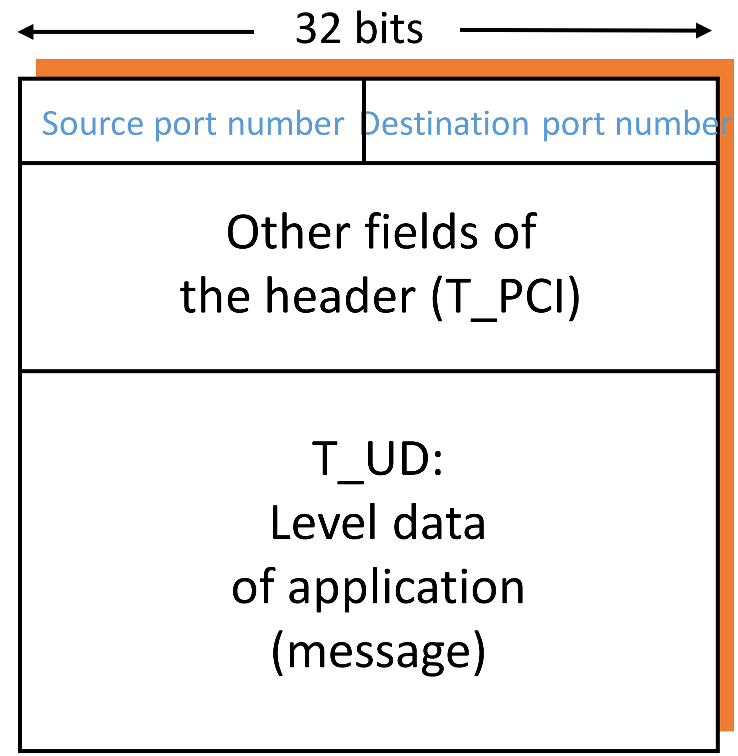
Deliver in the correct socket the content (T_UD) of the segments (T_PDU) received, thanks to the information in the header (T_PCI).



Multiplexing and Demultiplexing

Operation

- The network layer receives IP datagrams (N_PDU)
    - Each N_PDU has a source IP address and a destination IP address in its header (N_PCI).
    - Each N_PDU encapsulates a segment (T_PDU)[1].
- Transport layer receives segments (T_PDU)
    - Each T_PDU has in its T_PCI
    - a source port number and
    - a destination port number.
    - Each T_PDU encapsulates user, application-tier (T_UD) data.
    - **On the destination host, IP addresses and port numbers are used to deliver the T_UD of the T_PDU to the appropriate socket.**

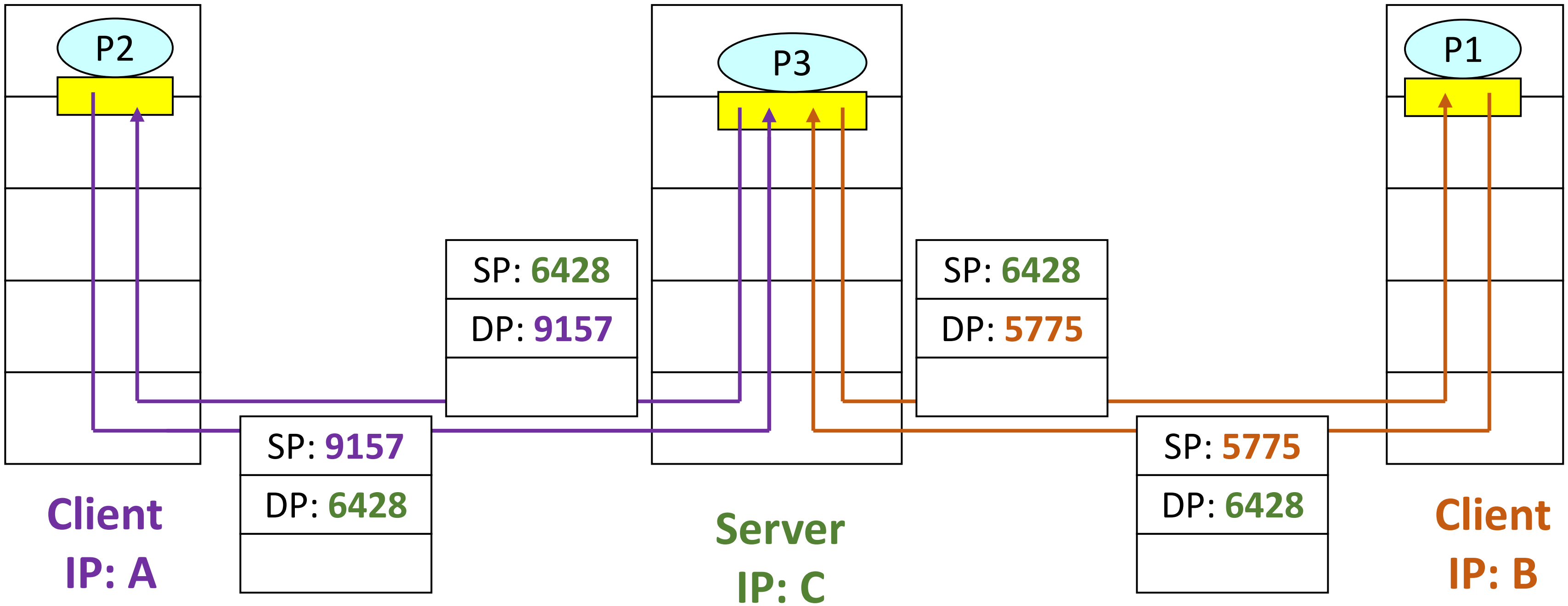


T_PDU format
(common to TCP and UDP)

1 True if there is no fragmentation

Multiplexing and Demultiplexing

Not connection (UDP)



The Source IP and Source Port will allow the P3 process to identify the source process (P1 or P2) and return a message to it.

SP = Source Port Number
DP = Destination Port Number

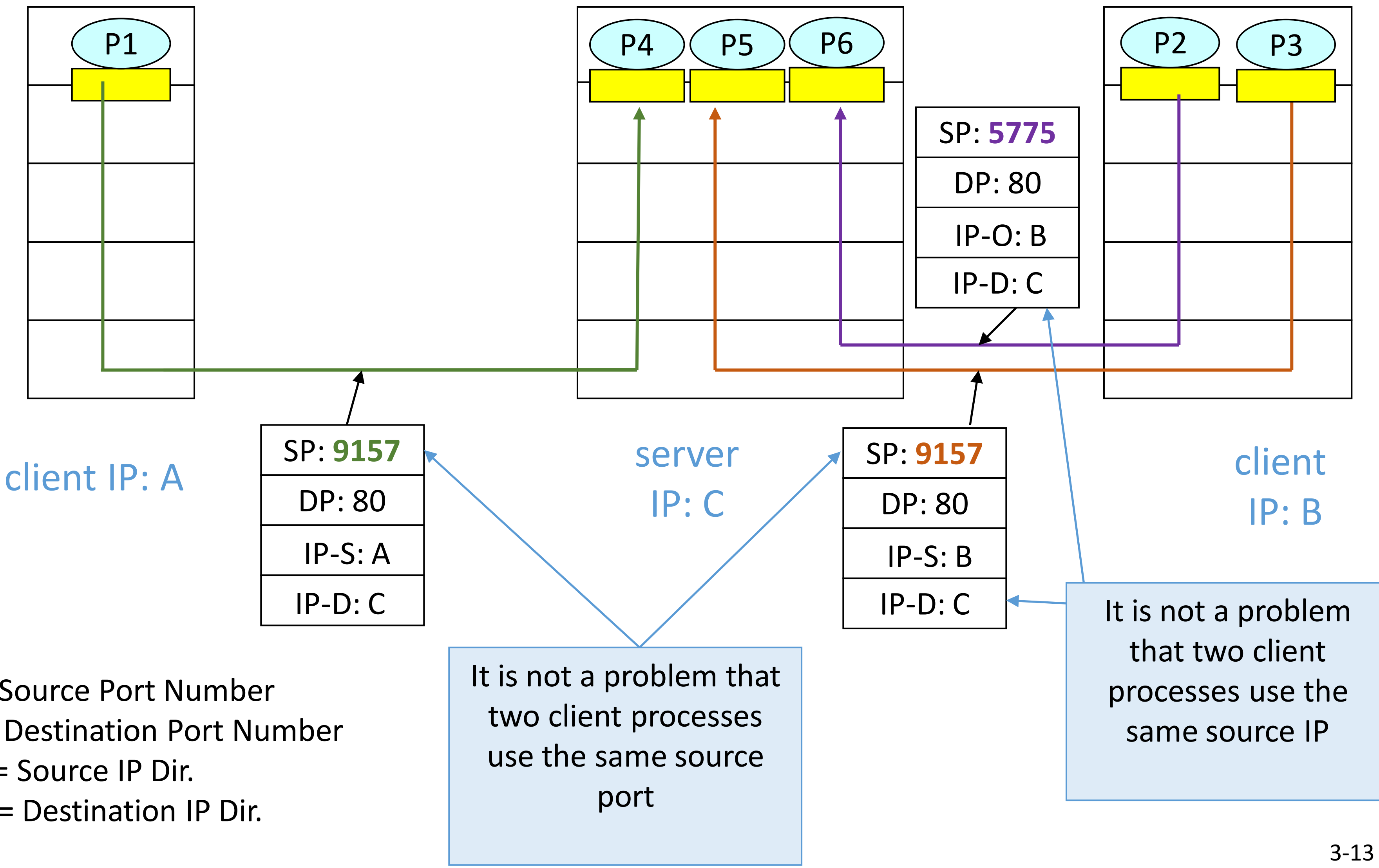
Multiplexing and Demultiplexing
Connection-oriented (TCP)

- A TCP connection is identified by a 4-tuple:
  - Local IP address
  - Local Port No.
  - Remote IP Address
  - Remote Port No.
- On the destination host, the 4 values (present in the N_PCI and T_PCI) are used to get the user data of the T_PDU to the appropriate TCP socket.
-

- a server application can have multiple tcp connections running simultaneously.
  - Each connection is identified by its own 4-tuple
  - a web server has a different tcp connection for each client that connects.
  - With non-persistent HTTP, each request from the same client will go for a different TCP connection.

Multiplexing and Demultiplexing

Web server with multiple processes



SP = Source Port Number
DP = Destination Port Number
IP-S = Source IP Dir.
IP-D = Destination IP Dir.

Lesson 3: The Transport Layer

Objectives

- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

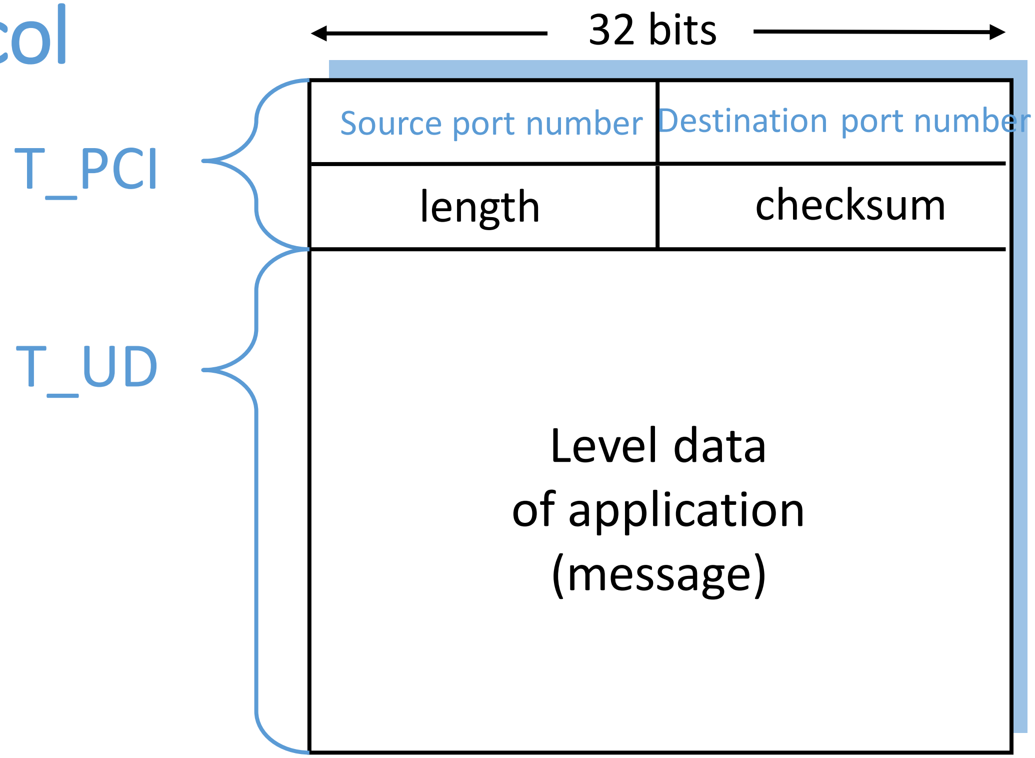
1. Transportation Level Services
2. Multiplexing and demultiplexing
3. **No conexion transport: UDP**
4. Principles of reliable transfer
5. Connection-oriented transport: TCP
   - TCP segment structure
   - Reliable data transfer
   - Flow control
   - Connection management

Non-conection oriented transport: UDP
UDP = User Datagram Protocol [RFC 768]

- It is a very simple Internet transport protocol.
- Offers a best effort service:
  - The T_PDU can be "lost" and not reach their destination.
  - If the T_PDU arrive in a different order, the user data contained in them would be delivered in a different order to the Application Level.
- **Non-conection:**
  - There is no pre-agreement phase between the UDP sender and receiver.
  - Each T_PDU is treated independently of the others.

User Datagram Protocol

- Often used by media streaming applications
- Loss tolerant
- Bandwidth sensitive
- Other uses of UDP
  - DNS
  - SNMP
- Reliable transfer over UDP? It is possible if we add reliability to the application layer
- Application-specific error recovery!



T_PDU format of UDP

The header (T_PCI) only has 4 fields.
The length is in bytes and is that of the full T_PDU, with header.

UDP
Checksum

.**Objective: to detect "errors" (e.g., "changed" bits) in a transmitted T_PDU**

The sender:

Treats the T_PDU as a sequence of 16-bit integers.

Simplifying a bit, we can say that it adds all the 16-bit integers that make up the T_PDU and then calculates the complement to 1.

Place the calculated value in the checksum field of the header (T_PCI).

The receiver:

It calculates the checksum, again, in the same way that the issuer did, on the T_PDU received.

Checks whether the calculated checksum is identical to the value of the checksum field of the received T_PDU.

NO: Error detected!

YES → No error is detected, but ... Could there be an error?

Lesson 3: The Transport Layer

Objectives

- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

1. Transportation Level Services
2. Multiplexing and demultiplexing
3. No conexion transport: UDP
4. **Principles of reliable transfer**
5. Connection-oriented transport: TCP
   - TCP segment structure
   - Reliable data transfer
   - Flow control
   - Connection management

Principles of reliable transfer

R.T. is important at the application, transport, and data link layers.

It's on the list of the 10 most important topics about networking!

Why is R.T. necessary?

Wrong PDUs

In the transmissions over the links there is interference that alters the transmitted bits.

Lost PDUs

Packet queues, when saturated, begin to discard incoming packets.

Duplicate PDUs

Certain communication problems cause PDUs that have already been received to be retransmitted.

How are these problems detected?

Wrong PDUs

Using error checking mechanisms (included in the PCI).

Algorithms similar to checksum.

The most complex and reliable algorithms are used at the link level.

Lost and duplicated PDUs

Adding "something" to the header (PCI) of each PDU that allows it to be distinguished from the rest of the sent PDUs.

How are these problems solved?

Retransmissions

The transmitter re-transmits an exact copy of the PDU that had problems.

Principles of reliable transfer
Communication between peer entities

General case of communication between peer entities of the same level:

- When communicating with your peer entity, one entity sends the other a header PDU (PCI) and, in general, user data (UD).
- **Headers (PCI) contain protocol control information.**
- In general, both entities transmit and receive user data.
- **Bidirectional transfer of user data between peer entities.**

Simplification of the general case of communication between peer entities:

It makes easier the explanation of the principles of reliable transfer.

We will assume a one-way transfer of user data.

- One of the peer entities of the level will be called **transmitter** (Tx)**.**
- We will call the other entity a **receiver** (Rx).
- The Tx transmits PDUs with PCI and UD (the UDs come from their upper level).
- The Rx receives PDUs with PCI and UD (the UD will pass them to their higher level).
- The Rx transmits PDUs that will only have PCI (no UD, only control info).
- The Tx receives PDUs that will only have PCI.
- **Bidirectional transfer of protocol control information.**

Principles of reliable transfer

Types of PDUs

- **Data PDU**
- Only the Tx sends them
- Contains user data (UD)
- Contains protocol control information (on the PCI)
- **Control PDU**
- They are only sent by the Rx
- **Contains no user data (UD)**
- Only contains protocol control information (in the PCI)

Note

It is a mistake to think that the Tx does not send control information because it only sends PDUs of data.

Data PDUs also carry control information.

Principles of reliable transfer
What does the header (PCI) contain?

- The **PCI** of a **data PDU** contains information that enables:
  - Let the Rx detect if that data PDU has errors.
  - Identify that data PDU and distinguish it from other PDU sent by the Tx.
- The PCI of a control PDU contains information that allows:
  - Let the Tx detect if that control PDU has errors.
  - That the Rx identifies a certain data PDU, and informs about that Data PDU
  - has been received correctly by the Rx.
      - ACK, acknowledgement
    - **has not been received correctly by the Rx**.
      - NAK, NACK, negative acknowledgement,

Note

the PCI information that is used to identify a particular data PDU is called a sequence number.

Principles of reliable transfer

Basic operation

Transmitter (Tx):

- The Tx entity of a certain level, builds a data PDU with UD and PCI and transmits it to the Rx.
- Now, the Tx waits for a while, known as time_out, to receive an Rx control PDU
- with an ACK means the data PDU arrived correctly to the Rx.
- Tx doesn't do anything else.
  - with a NACK, that the data PDU arrived with errors to the Rx.
  - Tx resend the PDU.
  - If the time_out expires before the Rx control PDU arrives, then
  - Tx resend the PDU.

Receptor (Rx):

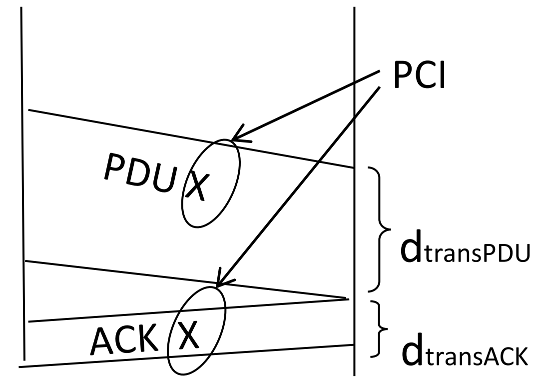
- When receiving a data PDU the Rx entity of a certain level:
  - If the data PDU arrived correctly, it is mandatory for the Rx to send the TX an ACK type control PDU.
  - If the data PDU arrived with errors, it is optional for the Rx to send the TX a control PDU of type NACK.
  -

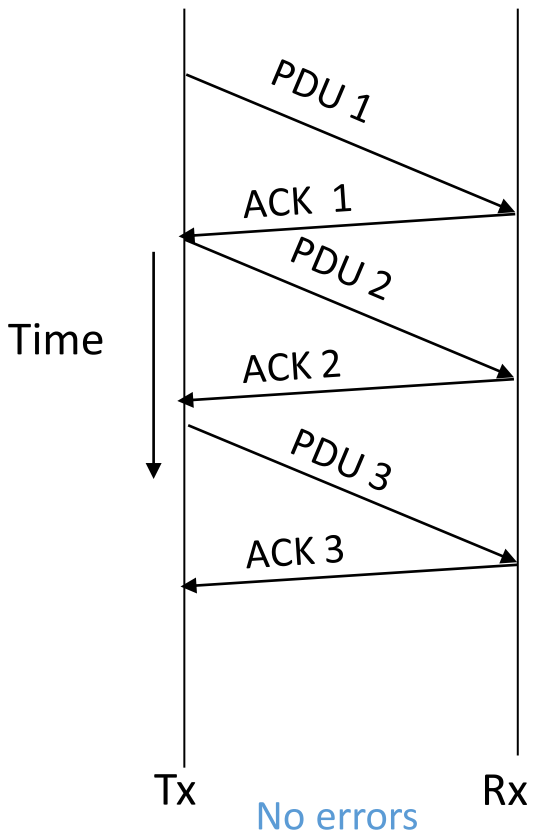
P: How much should the time_out be, at a minimum?

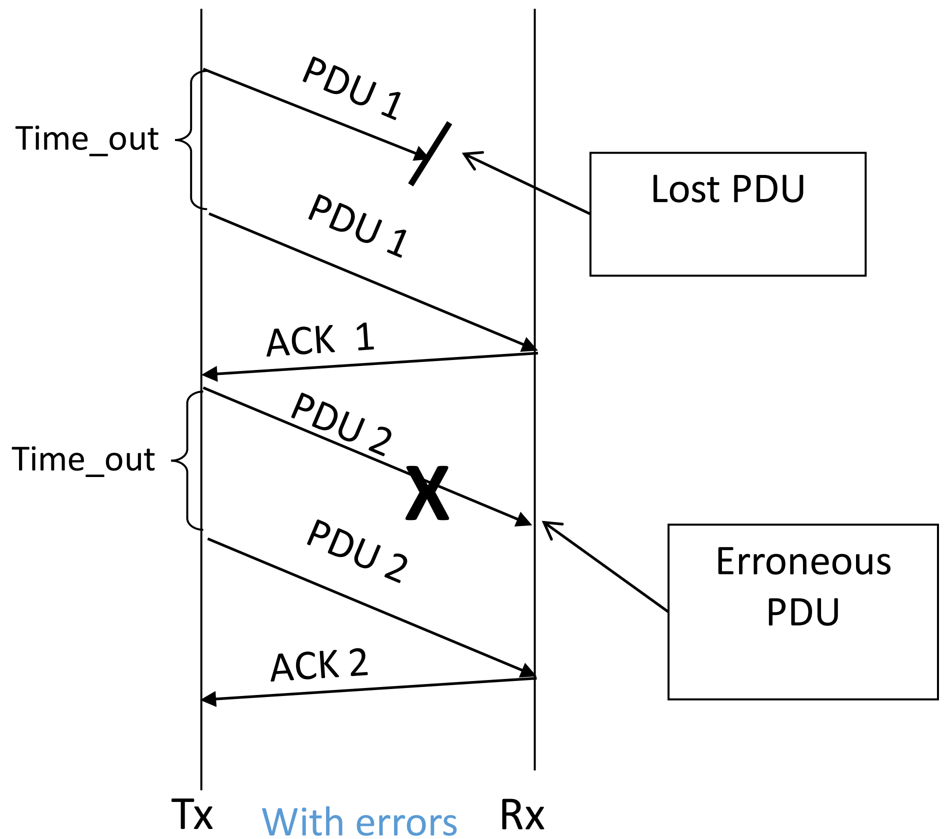
P: Does the Rx always deliver the UD of a data PDU that arrives correctly to the top level?

Principles of reliable transfer
Example 1: loss and error

- Transmitter sends only one PDU with UD and does not send the next one until successful transmission.
- Receiver only sends ACK control PDU.



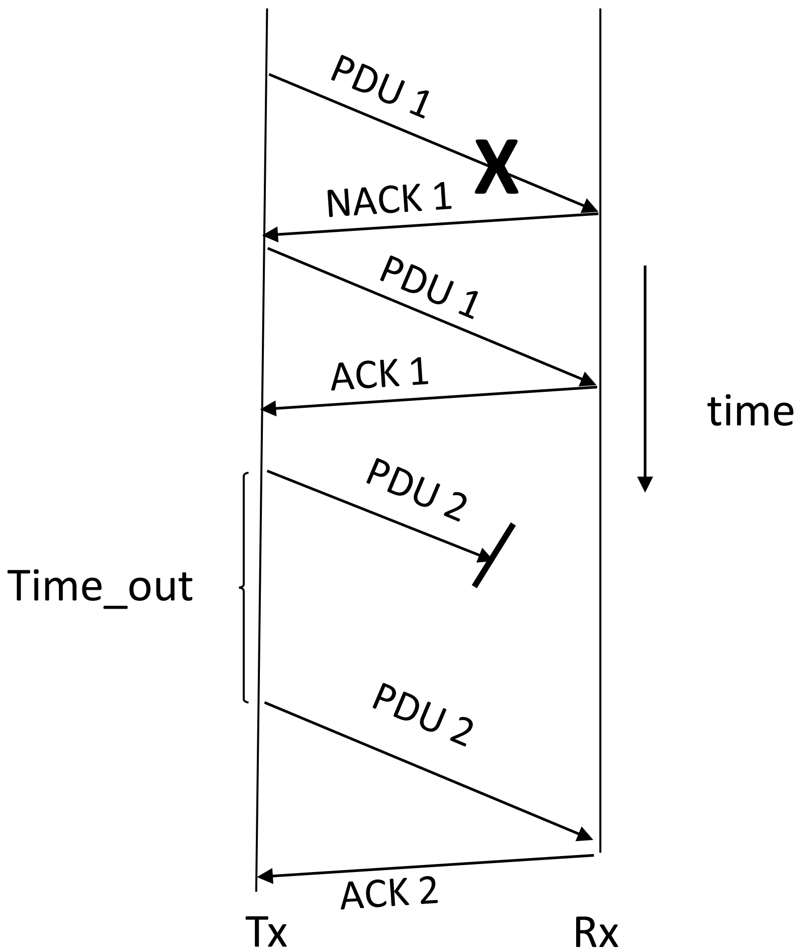




Principles of reliable transfer

Example 2: Negative acknowledgement (NACK)

- Idem example 1.
- Receiver also sends NACK control PDU.
- No erros.
- same previous behavior



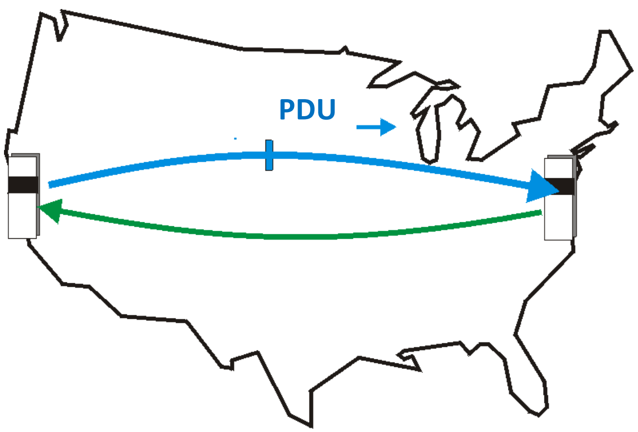
With errors

Principles of reliable transfer
Protocols with Pipeline - Concept

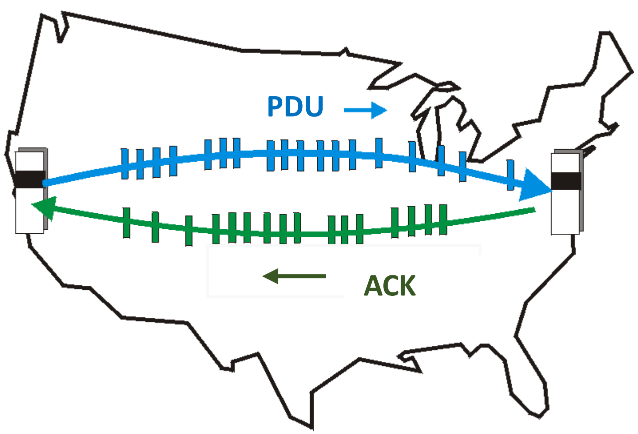
With pipeline, the Tx can have multiple PDUs in transit ("in flight") pending confirmation (ACK), greatly improving efficiency.

The sequence numbers used in the PCI must be of sufficient range to distinguish all PDUs in transit.

Buffers are required in the Tx and sometimes in Rx.



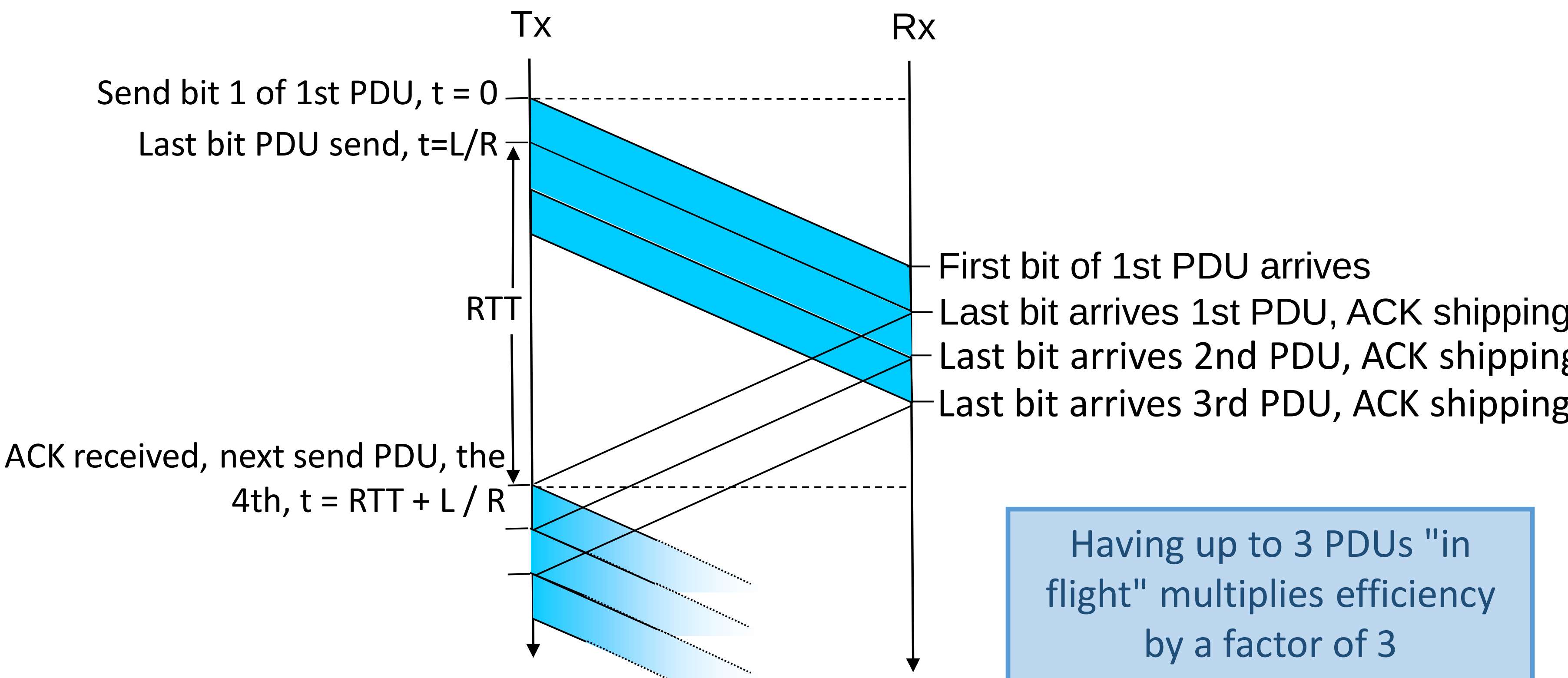
Stop and wait



Pipeline

Principles of reliable transfer

Protocols with Pipeline - Improving Efficiency



Having up to 3 PDUs "in flight" multiplies efficiency by a factor of 3

$$U_{transmisor} = \frac{3 * L / R}{RTT + L / R} = \frac{0,024}{30,008} = 0,0008$$

Lesson 3: The Transport Layer

Objectives

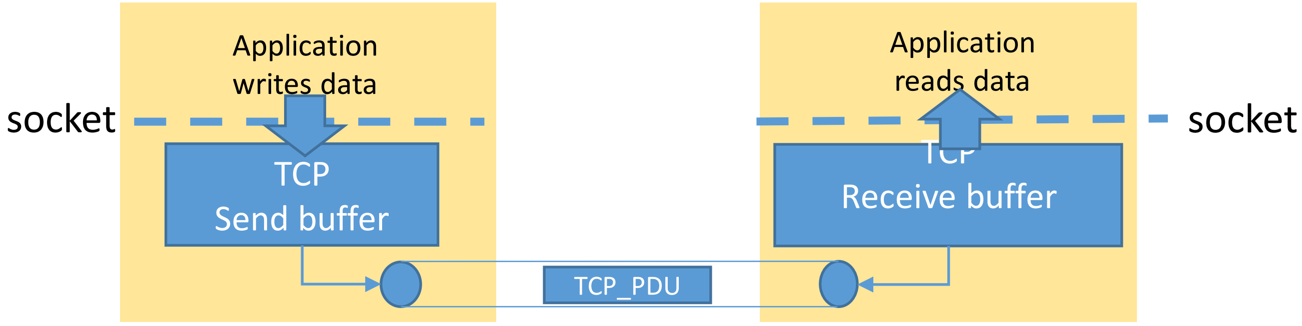
- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

1. Transportation Level Services
2. Multiplexing and demultiplexing
3. No conexion transport: UDP
4. Principles of reliable transfer
5. **Connection-oriented transport: TCP**
   - **TCP segment structure**
   - Reliable data transfer
   - Flow control
   - Connection management

Connection-oriented transport: TCP
TCP Segment Structure – Overview

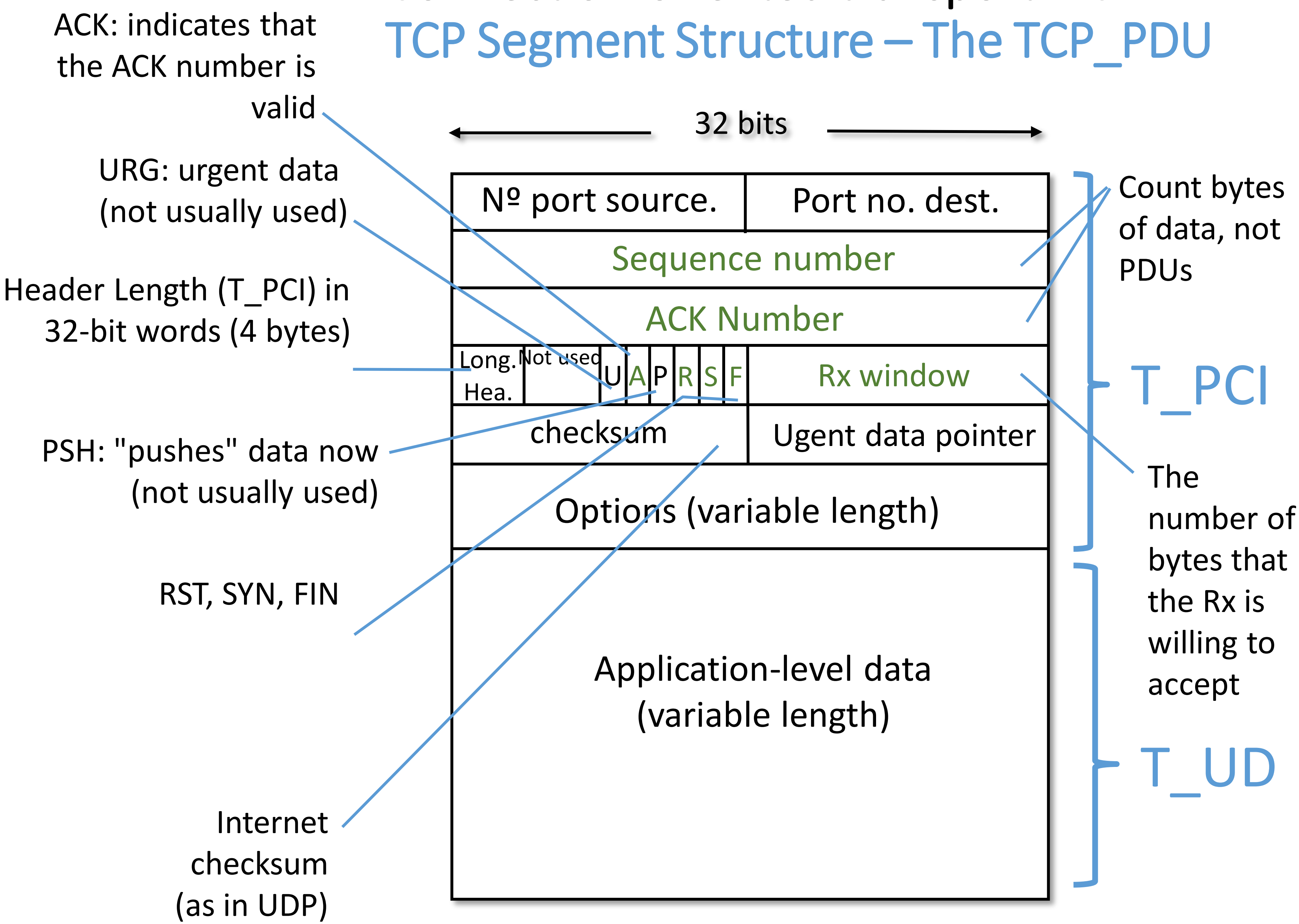
- **RFCs: 793, 1122, 1323, 2018, 2581**
- Point to point:
  - One sender, one receiver (non-multicast)
- Byte flow, reliable and in order:
- No "border" between messages (A_PDUs)
- Use "pipeline":
- TCP flow control sets the size of the window (max. no data "in flight")
- Buffers: Btx y Brx



- Full-Duplex:
  - Data flows through a connection bidirectionally.
- MSS: Maximum Segment Size (actually, of the T_UD). It is negotiated when the connection starts.
- Connection-oriented:
- Agreement prior to sending data. The client takes the initiative by sending a control message, which the server should be waiting for.
- Flow control

Connection-oriented transport: TCP

TCP Segment Structure – The TCP_PDU



Connection-oriented transport: TCP

TCP Segment Structure – Sequence Number and ACK

Sequence number:

It is the number assigned, within the byte stream, to the first byte of tcp segment data that is sent to the other peer entity.

The initial value of this field is decided randomly by each peer entity at the start of the connection.

It increases as segments containing UD are sent.

ACK No.:

It is used to indicate the sequence number of the byte that is expected to be received next by the other even entity.

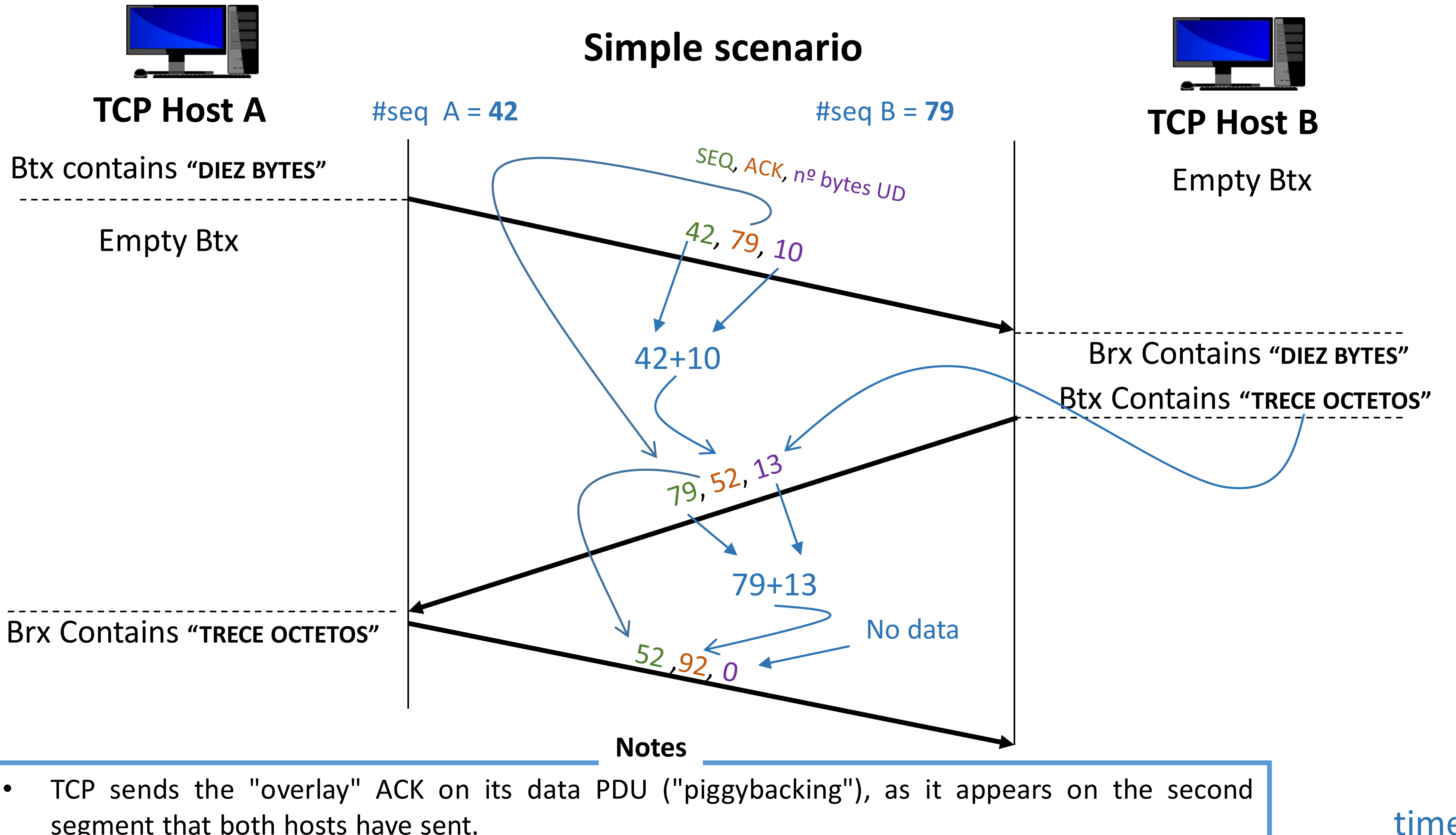
All previous bytes are recognized (cumulative ACK).

Q: How does the receiver treat out of order segments?

A: The TCP specification leaves it to the discretion of the implementer.

Connection-oriented transport: TCP

TCP Segment Structure – Sequence Number and ACK – Example



Notes

- TCP sends the "overlay" ACK on its data PDU ("piggybacking"), as it appears on the second segment that both hosts have sent.

Lesson 3: The Transport Layer

Objectives

- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

1. Transportation Level Services
2. Multiplexing and demultiplexing
3. No conexion transport: UDP
4. Principles of reliable transfer
5. Connection-oriented transport: TCP
   - TCP segment structure
   - **Reliable data transfer**
   - Flow control
   - Connection management

Connection-oriented transport: TCP

Reliable Data Transfer – Events Handled by the TCP Sender (simplified)

Application data arrives (top level):

- Creates a segment with an appropriate sequence number and passes it to the lower level (IP).
- The sequence number of the segment is the sequence number, within the byte stream, of the first byte of data in the segment.
- Start the timer, if it was not already running (it would be running if there was previous data not recognized).
- The timer is set to expire after Time_out seconds.

Timer expires (time_out):

It broadcasts the segment that has caused the time_out.

Reboot the timer.

An ACK arrives...

... for data for which an ACK had not yet been received:

Updates the indicator that points to the "oldest pending ACK data" and stops the timer.

Start the timer only if there is still "in flight" pending data from ACK.

TCP estimates the RTT to know what value to use in time_out

What value should TCP use as a time_out?

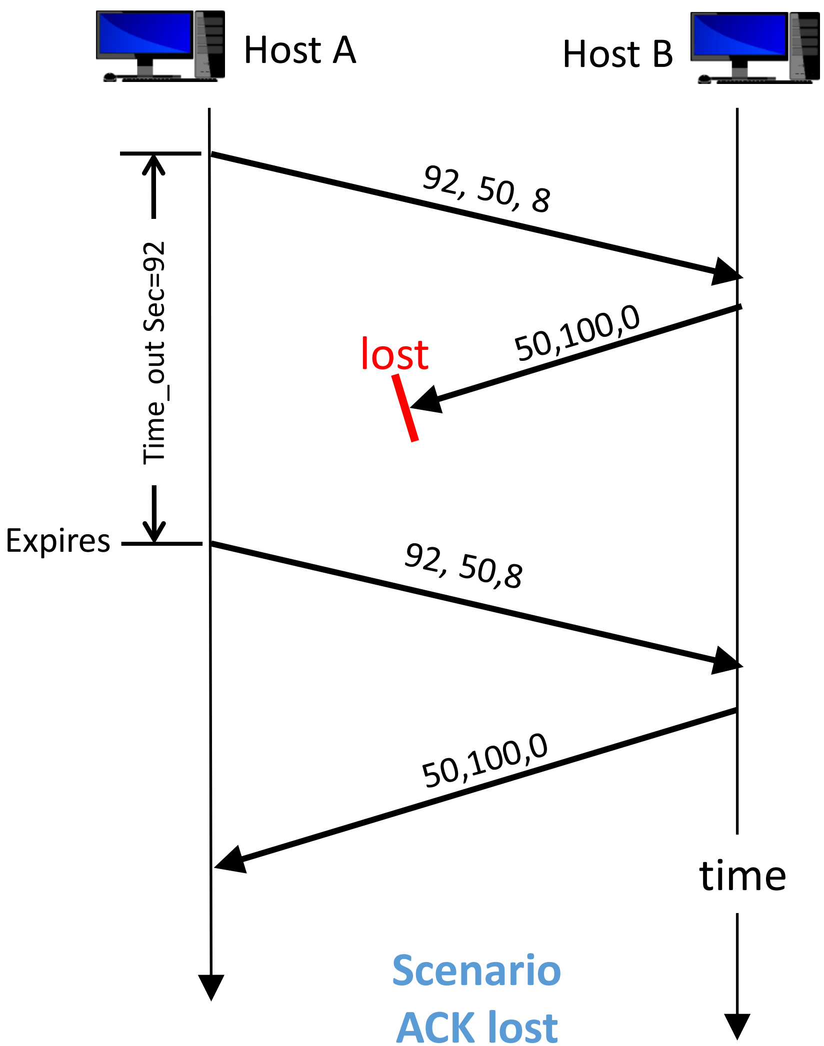
Must be somewhat larger than RTT

- But RTT changes over time...
- A very small value produces premature time_out and unnecessary retransmissions.
- Too large a value causes you to react too late to the loss of a segment.

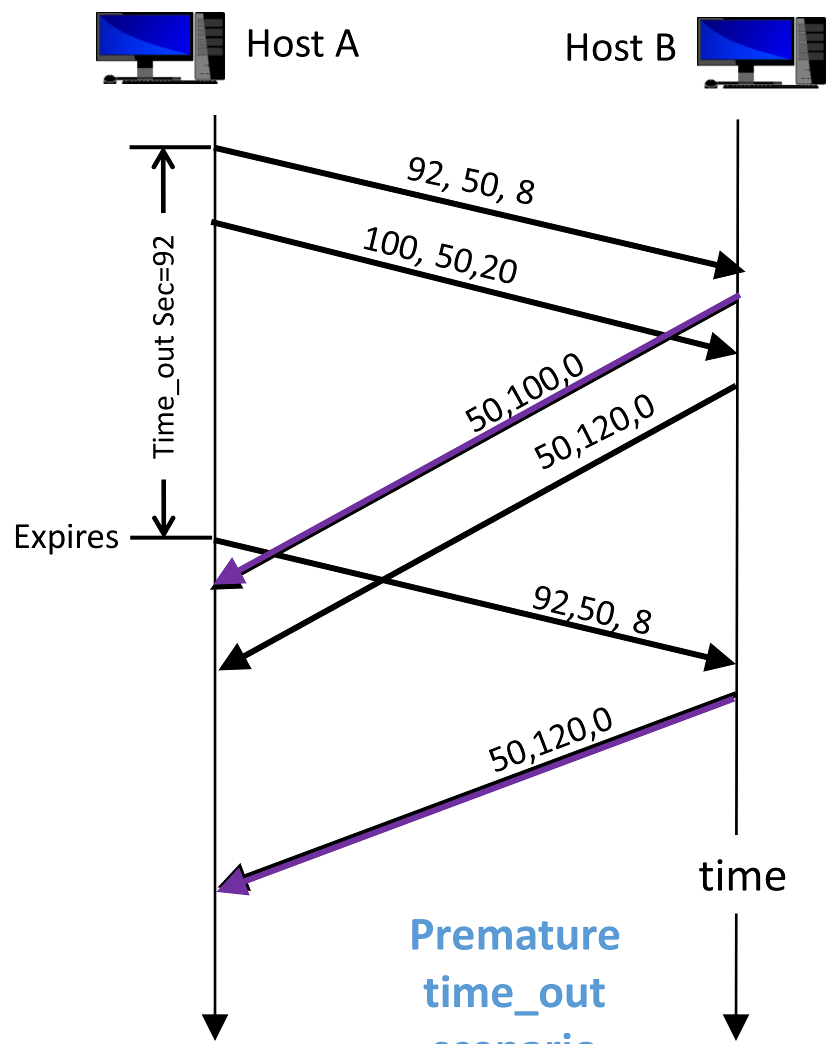
TCP follows an algorithm to estimate, in real time, the RTT that exists at each moment and then calculates the timeout

Connection-oriented transport: TCP

Reliable data transfer – Scenarios with relays (I)



Scenario ACK lost

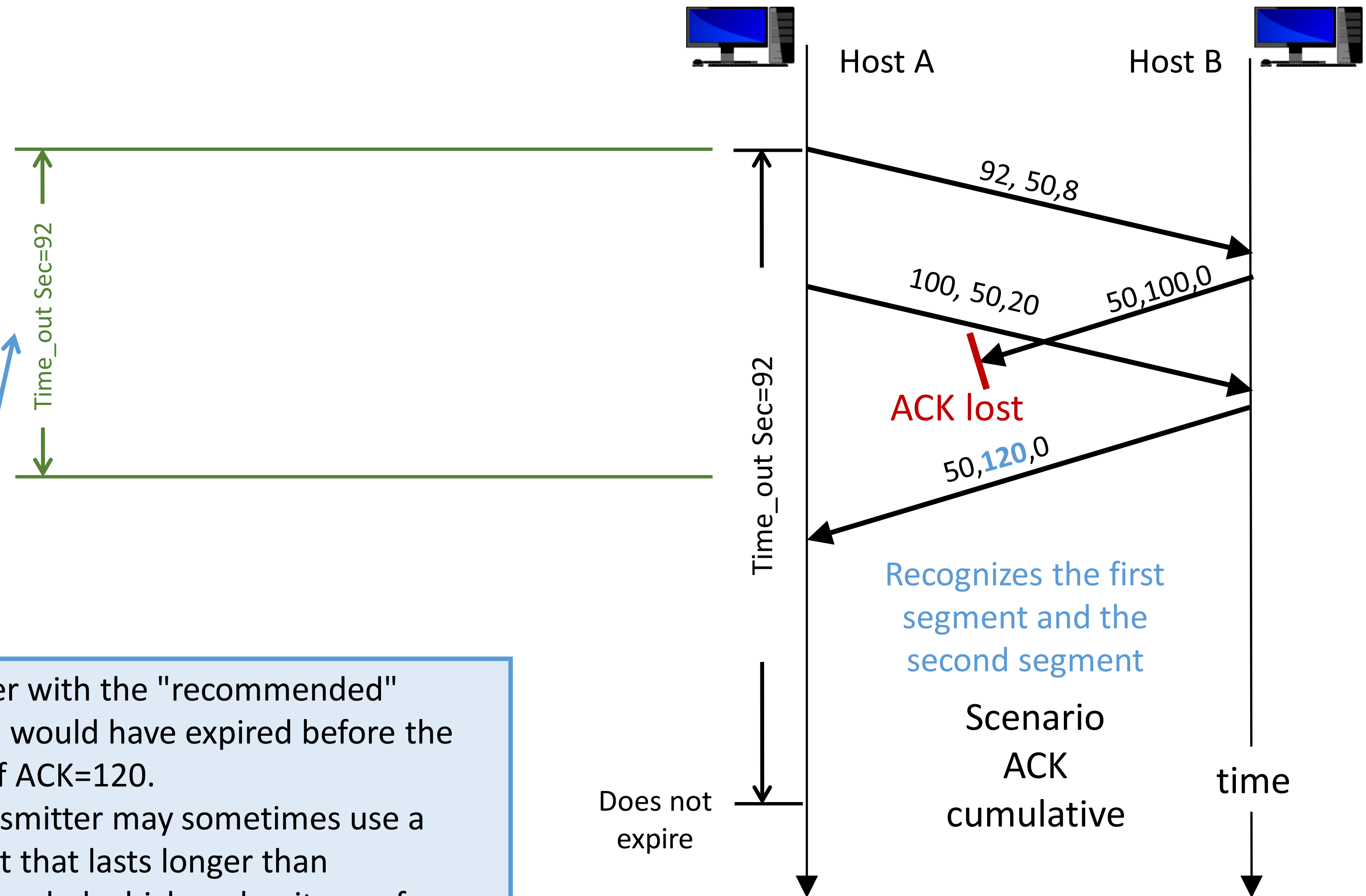


Premature time_out scenario

Duplicates arrive but TCP only goes up one to the Application level

Connection-oriented transport: TCP

Reliable data transfer – Scenarios with relays (II)



The timer with the "recommended" duration would have expired before the arrival of ACK=120.
The transmitter may sometimes use a time_out that lasts longer than recommended which makes it easy for cumulative ACKs to occur.

Connection-oriented transport: TCP

Reliable data transfer – Fast retransmission (II)

- The time_out has a relatively long duration:
  - It takes a long time for a lost segment to be broadcast.

Detect lost segments thanks to duplicate ACKs.

The sender often sends many segments in a row, very “close".

If a segment is lost, many duplicate ACKs will most likely arrive.

If the sender receives three duplicate ACKs for the same data, it assumes that the segment whose data follows the data being recognized has been lost:

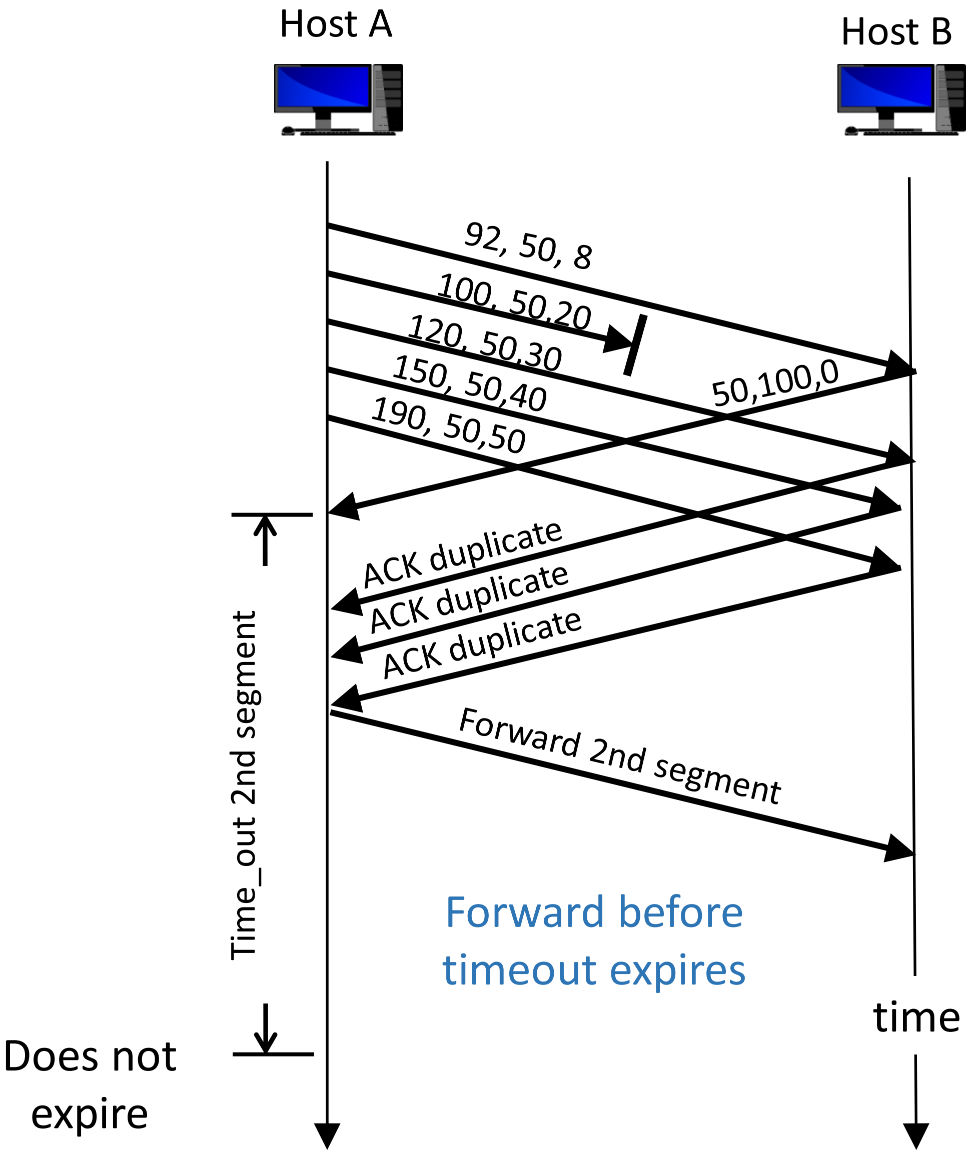
Fast retransmission: Forward that segment that is supposed to be lost even though its timer has not yet expired.

Note

TCP does not use NAKs, so the receiver cannot warn that a segment is missing.

Connection-oriented transport: TCP

Reliable data transfer – Fast retransmission



Lesson 3: The Transport Layer

Objectives

- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

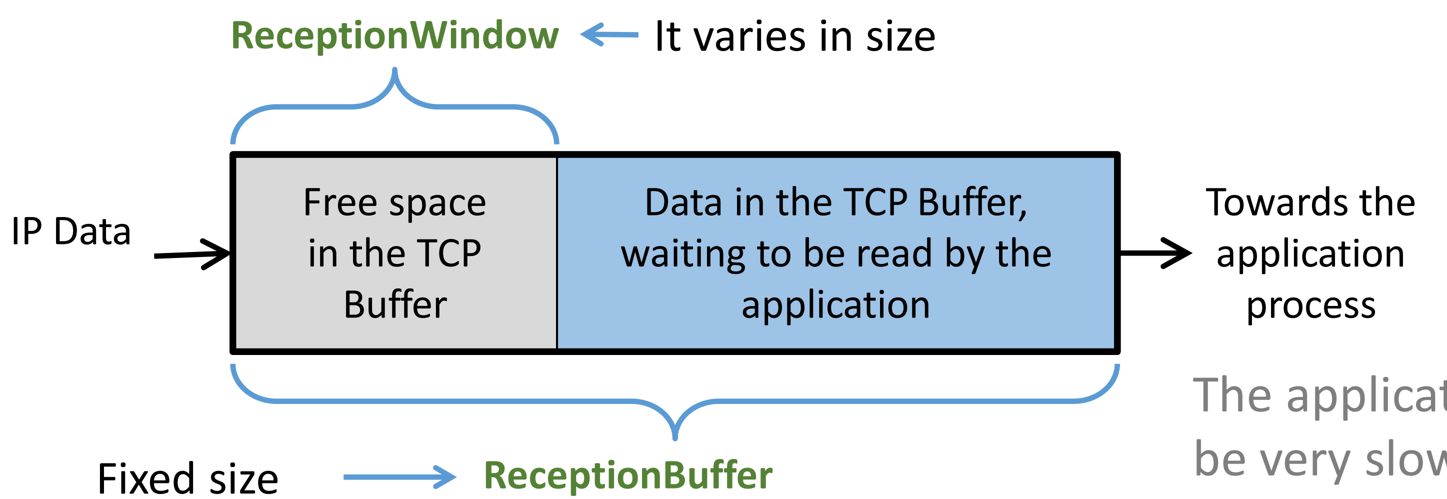
1. Transportation Level Services
2. Multiplexing and demultiplexing
3. No conexion transport: UDP
4. Principles of reliable transfer
5. Connection-oriented transport: TCP
   - TCP segment structure
   - Reliable data transfer
   - **Flow control**
   - Connection management

Connection-oriented transport: TCP
Flow control

Flow control

Getting the sender not to overflow the receiver buffer because of transmitting too much data, too fast.

- The receiving side of a TCP connection has a receive buffer where data arriving from the lower level accumulates.
-

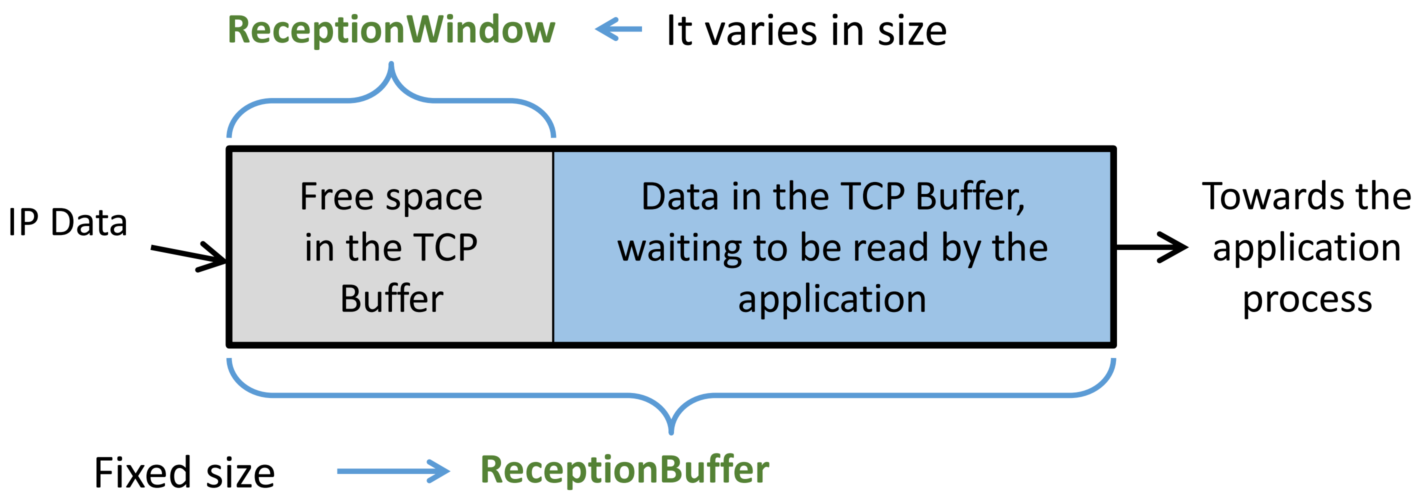


The application process may be very slow reading from the TCP receive buffer.

- Speed adjustment service:
- It makes the transmission rate at the other end adjust to the rate at which the receiving application "consumes" the data that arrives.

Connection-oriented transport: TCP

Flow Control – Operation (I)

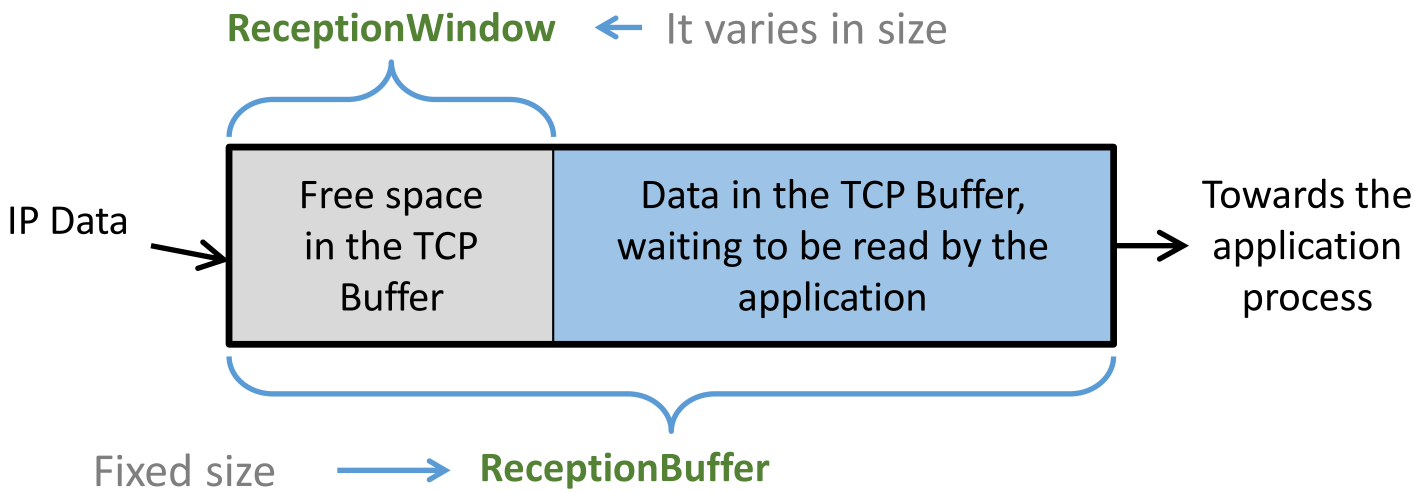


- We will assume that the TCP receiver discards the segments it receives disordered, so those do not take up space in the Rx buffer.
- Free space in the receive buffer:

ReceptionWindow = **ReceptionBuffer** – (**LastByteReceived** – **LastByteRead**)

This difference indicates what is "occupied"

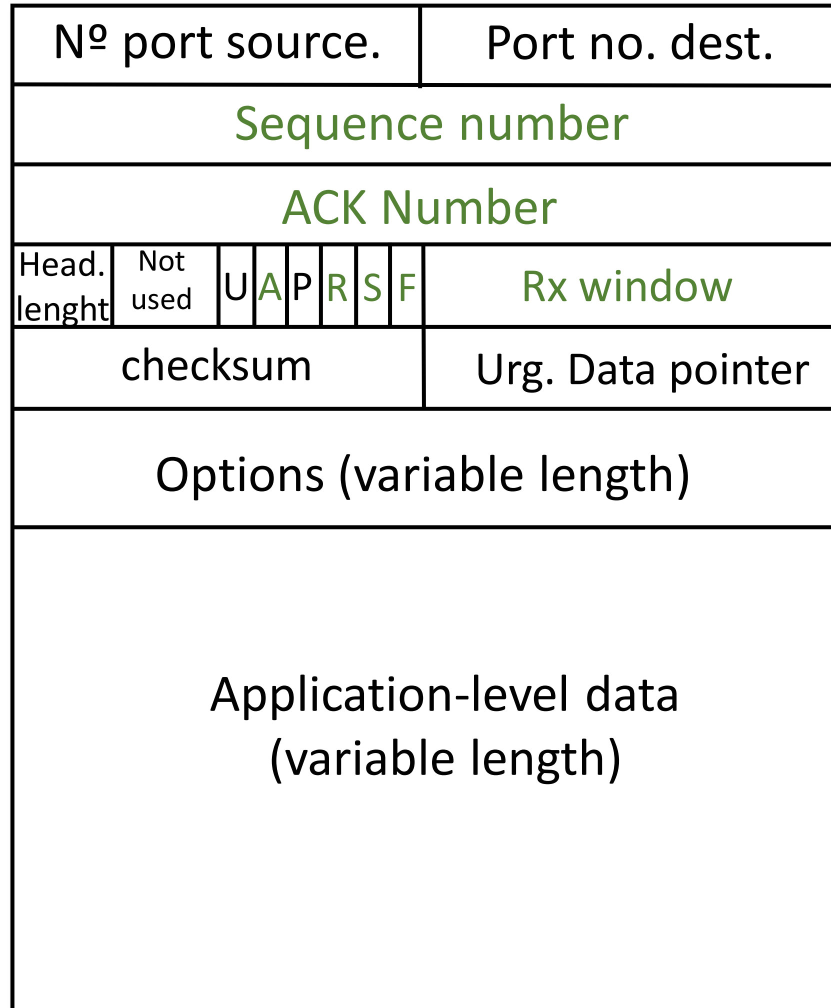
Connection-oriented transport: TCP
Flow Control – Operation (II)



- The receiver informs the sender of the free space in the buffer thanks to the ReceptionWindow field present in the header (T_PCI) of the segments (T_PDU) it sends.
- The sender limits the number of pending ACK "in-flight" data so that it fits in the Perception Window.
- This ensures that the ReceptionBuffer never overflows.

Connection-oriented transport: TCP
Flow Control – Operation (III)

- The value of the ReceptionWindow field in the T_PCI is related to the value of the ACK number field in the T_PCI.
- When the sender receives a segment, it observes the value of the ACK number field and the value of the ReceptionWindow field to know the range of sequence numbers corresponding to the data that it is authorized to have "in flight", pending confirmation:



Desde **ACK number** hasta (**ACK number** + **ReceptionWindow** – 1)

Lesson 3: The Transport Layer

Objectives

- Understand the principles behind transportation level services:
- Multiplexing/demultiplexing
  - Reliable data transfer
  - Flow control
- Know the transport protocols used on the Internet:
- UDP: non-connection-oriented transport
- TCP: Connection-Oriented Transport

Content

1. Transportation Level Services
2. Multiplexing and demultiplexing
3. No conexion transport: UDP
4. Principles of reliable transfer
5. Connection-oriented transport: TCP
   - TCP segment structure
   - Reliable data transfer
   - Flow control
   - **Connection management**

Connection-oriented transport: TCP
TCP Connection Management: Establishment

Remember: The TCP client and server establish the connection before exchanging segments that carry user data.

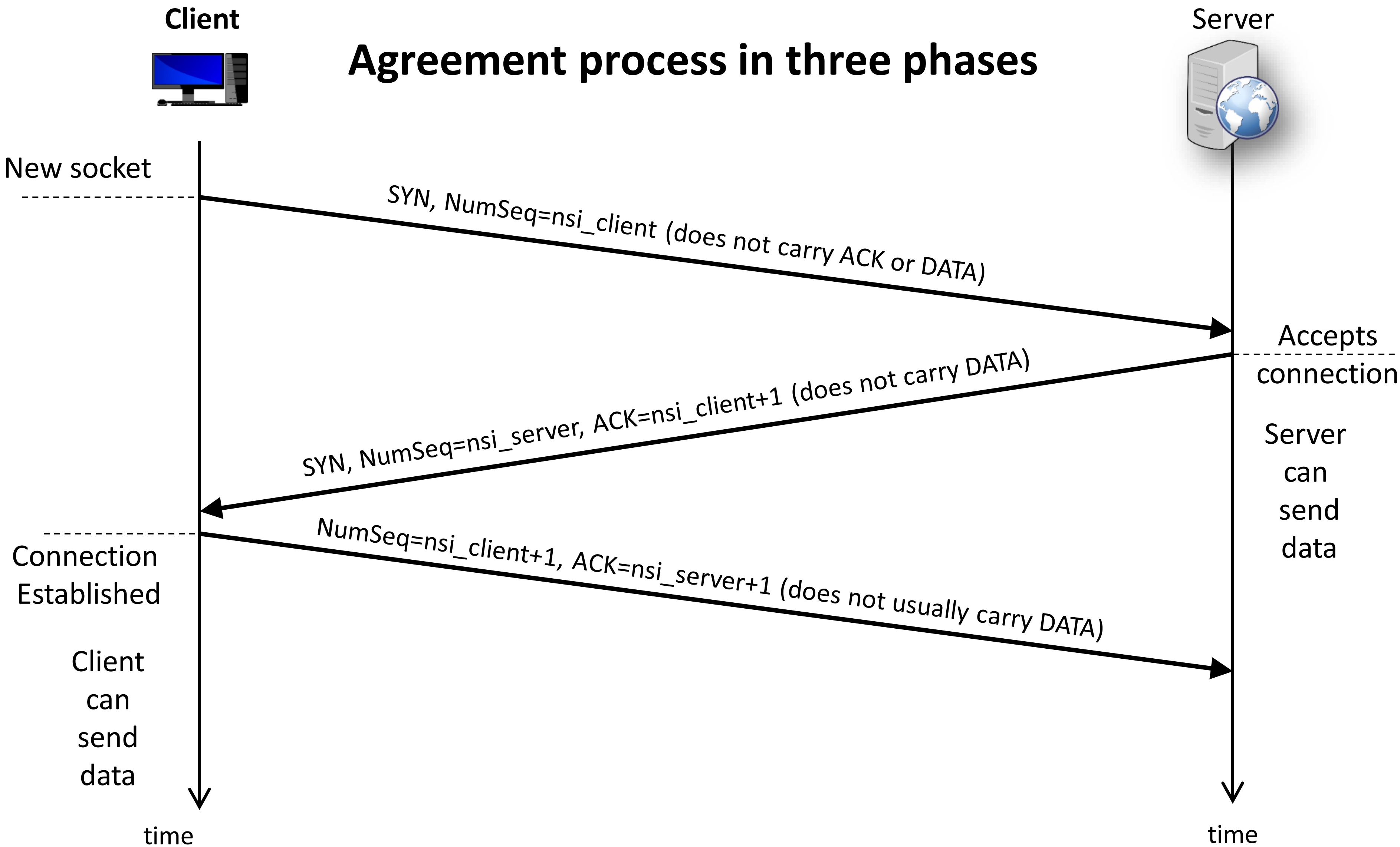
During this connection establishment phase both must initialize the tcp variables:

- Sequence numbers to use.
- Transmission and reception buffers (both on client and server).
- Flow control information (ex: **ReceptionWindow**).
- etc.

The client is the one who takes the initiative to establish the connection through a socket and the server is always "listening" to receive the connection start from the client.

Connection-oriented transport: TCP

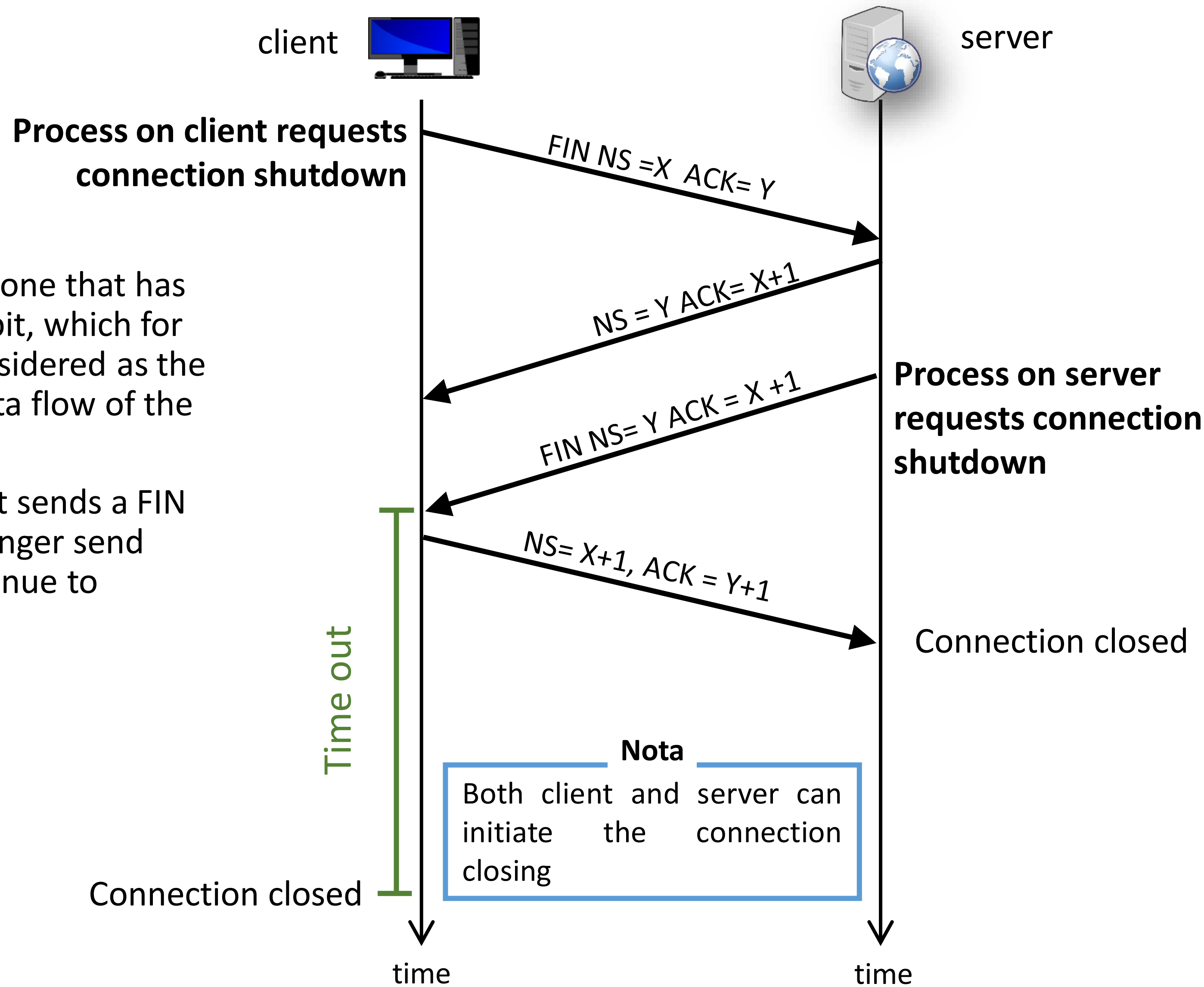
TCP Connection Management: Establishment



Connection-oriented transport: TCP

Connection Management: Closing the Connection TCP

- An FIN segment is one that has activated the FIN bit, which for all purposes is considered as the last byte of the data flow of the TCP connection.
- The TCP entity that sends a FIN segment can no longer send data, but can continue to receive it.



Contenidos

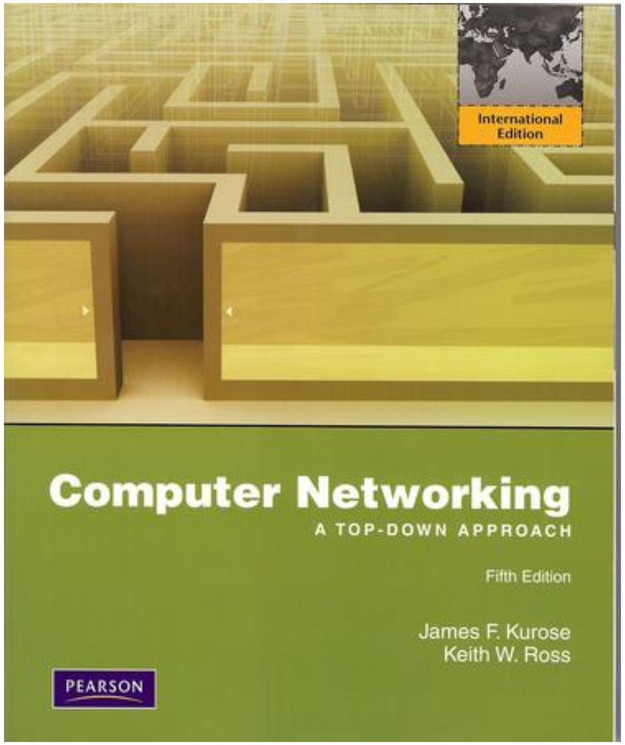
Tema 1: Redes de Computadores e Internet

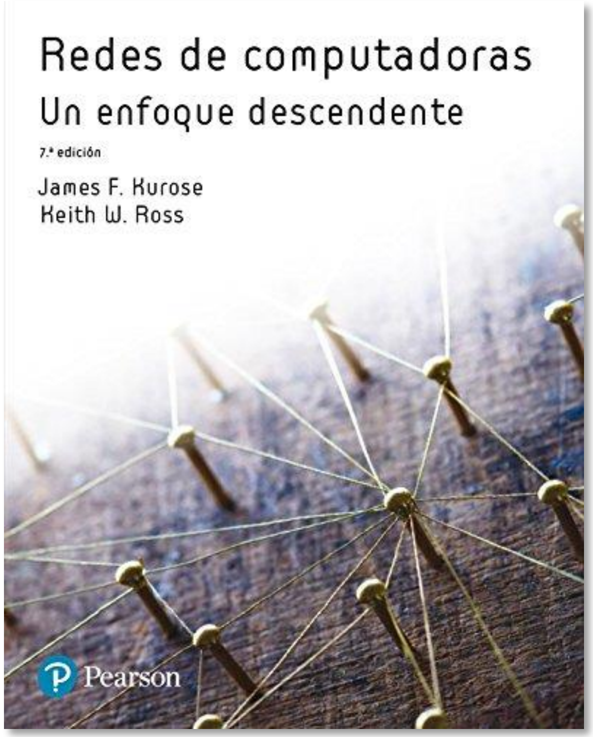
Tema 2: Capa de Aplicación

Tema 3: Capa de Transporte

Tema 4: Capa de Red

Tema 5: Capa de Enlace de Datos





Estas transparencias han sido elaboradas a partir de material con copyright que Pearson pone a disposición del profesorado, a partir del libro:

Jim Kurose, Keith Ross (2010). Computer Networking: A Top Down Approach, 5th edition, Ed. Pearson.

Algunas actualizaciones pertenecen a la última edición:

Jim Kurose, Keith Ross (2017). Redes de Computadoras: Un enfoque descendente, 7ª edición, Ed. Pearson.

Redes de Computadores
Tema 3

La Capa de Transporte

EJERCICIOS

Departamento de
Tecnología Electrónica

Problem 1

- Assume that client A initiates a TCP connection to a Web server named S. more or less simultaneously client B also initiates a TCP connection to S.
- Enter possible source and destination port numbers for:
  - Segments sent from A to S
  - Segments sent from B to S
  - Segments sent from S to A
  - Segments sent from S to B

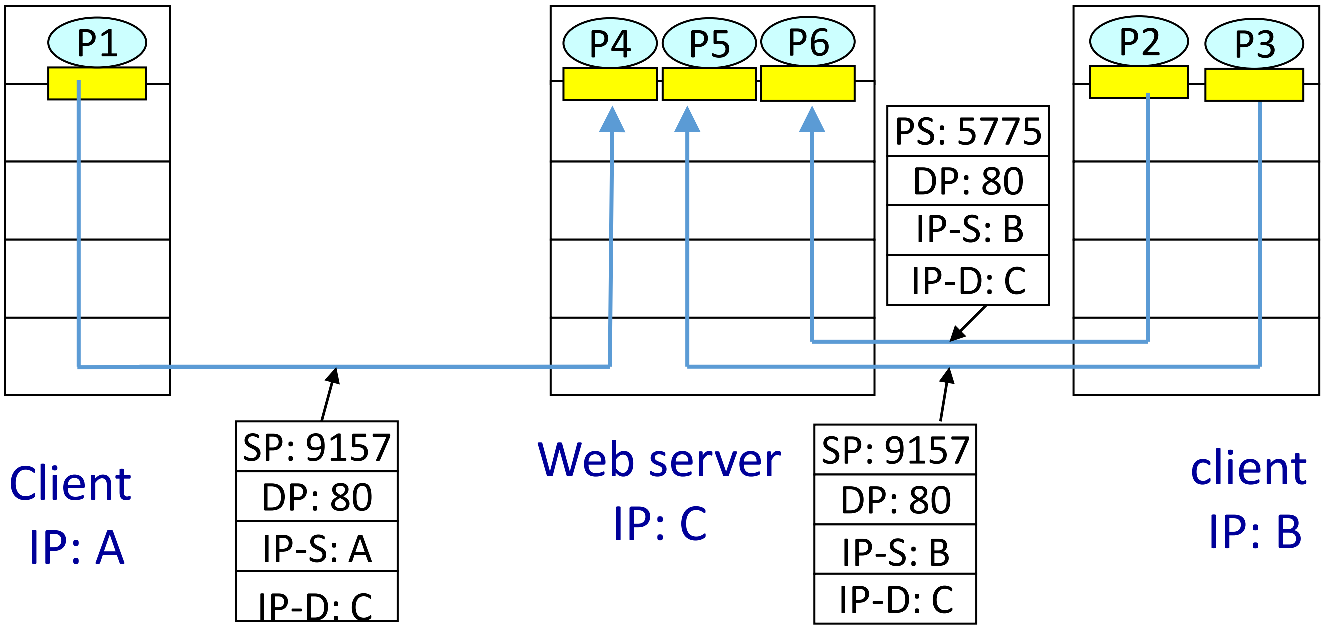
If A and B are on different hosts, could the source port number of segments from A to S be the same as that of segments from B to S?

What if client processes A and B are on the same host?

Problem 2

Observe the connections that clients have initiated to the Web server, and respond to the following:

SP = Source Port Number
DP = Destination Port Number
IP-S = IP Source
IP-D = IP Destination



a) What are the values of the source and destination ports on the segments that flow from the server back to the client processes?

b) What are the IP addresses (source and destination) of the network layer (N_PDU) datagrams that carry to those transport layer segments?

Problem 3

We have seen that pipeline protocols improve efficiency versus "stop and wait" protocols.

Suppose a Tx and an Rx a 1Gbps link, 30ms of RTT, with PDUs of 1500 bytes and with zero header sizes (that is, a totally negligible size compared to the other sizes).

How many data PDUs does the Tx have to be "in flight" for the channel utilization rate to be 95%?

Problem 4

An application may prefer UDP as a transport protocol over TCP, in order to have a greater degree of control over what data is sent on the T_PDU and at what time.

- Explain why UDP gives your application more control over what data is sent on the T_PDU.
- Explain why UDP gives your application more control over when a T_PDU is sent.

Problem 5

Assume that a client application protocol wants to send only a 1000-byte PDU using the TCP protocol to a server application that responds to it with 100 bytes.

Make a diagram with the flow of TCP_PDUs that will be exchanged by labeling each of them with the active flags, the value of the sequence number field, the value of the ACK number field, and the number of bytes of TCP_UD it carries. For each TCP_PDU exchanged, you must indicate the size in bytes of the exchange. assume tcp has no options the initial sequence number (nsi) of the client is 1000 and the initial sequence number (nsi) of the server is 3000.

Problem 6

A large file (L bytes) is to be transferred from A to B over a TCP connection on which the MSS has been set to 536 bytes. Calculate the maximum value that L can have if we do not want the sequence numbers used in the connection to start repeating.

Calculate the time it would take for a file of the L length you calculated earlier to be transmitted, assuming that:

- A and B are connected by a 155Mbps link
- Each TCP segment is encapsulated in a single IP datagram and this in a single frame, adding a total of 66 bytes of headers to the application-tier data.
- It can be sent at maximum rate, without danger of overflowing the receiver, so we will not take into account flow control.

Problem 7

Assume that a client application protocol wants to send only a 100-byte PDU using the TCP to a server application that responds to it with 1000 bytes.

Make a diagram with the flow of TCP_PDUs that will be exchanged by labeling each of them with the active flags, the value of the sequence number field, the value of the ACK number field, and the number of bytes of TCP_UD it carries. For each TCP_PDU exchanged, you must indicate the size in bytes of the exchange. assume that tcp has no options the initial sequence number (nsi) of the client is 0, the initial sequence number (nsi) of the server is 0, and the sss number is 536.

Problem 8

Hosts A and B are communicating over a TCP connection. Host B has received from A all bytes up to byte 126 and A has received its ACKs.

Suppose A now sends B two segments in a row, the first 70 bytes of data and the other 50 bytes.

The sequence number of the first segment is 127, the port of origin is 302 and the destination port number is 80.Suppose B sends an acknowledgment every time a segment of A arrives.

- What Sequence No., Origin Port No, and Destination Port Number does the second segment that sent A have?
- If the first segment reaches B before the second, what is the recognition number, origin port number and destination port number of the ACK that B will send through it?

Problem 8

c) If the network layer messes up the two segments of A, so that the second one arrives first at B, what will be the recognition number of the ACK that B will send as soon as it receives it?

d) Suppose that the two segments from A to B arrive in order, and that the two ACKs sent by B…..
   - The first one is lost and does not reach A
   - The second comes after the time_out of the first segment has occurred in A.

e) Draw a time diagram with the two segments sent by A, the two ACKs of B, and add the rest of the segments that are going to be sent due to retransmissions and new ACKs. Assume that no more segments will be lost. Enter data size, sequence number, and recognition number.