Curtin University

Transport Layer II

Prof. Ling Li | Dr. Nadith Pathirage | Lecture 07

Semester 1, 2021

Curtin University

Flow and **Congestion** Control

- Flow Control
  - Transport Layer
  - Data Link Layer
- Congestion Control
  - Warning-bit
  - Choke-packets
  - Load-shedding
  - RED

Flow Control – Data Link Layer

Curtin University

- Common flow and error control **techniques at the** **Data Link Layer:**

1. **Stop-and-Wait** ARQ
2. **Go-back-N** ARQ
3. **Select-Reject** ARQ

Sliding Window Methods

Flow Control – Transport Layer

Curtin University

- **Why limit flow?**
  - Source may send frames/packets at a rate that is faster than the processing speed of the destination host
  - The buffer at the destination could be full because
    - ✓ Higher level protocol is not ready
    - ✓ Outgoing I/O port not ready
    - ✓ Destination protocols cannot process PDU as fast

Prevents
Buffer
Overflow

"limits the amount of data being transmitted so that destination host can 'cope' "

Flow Control – Transport Layer

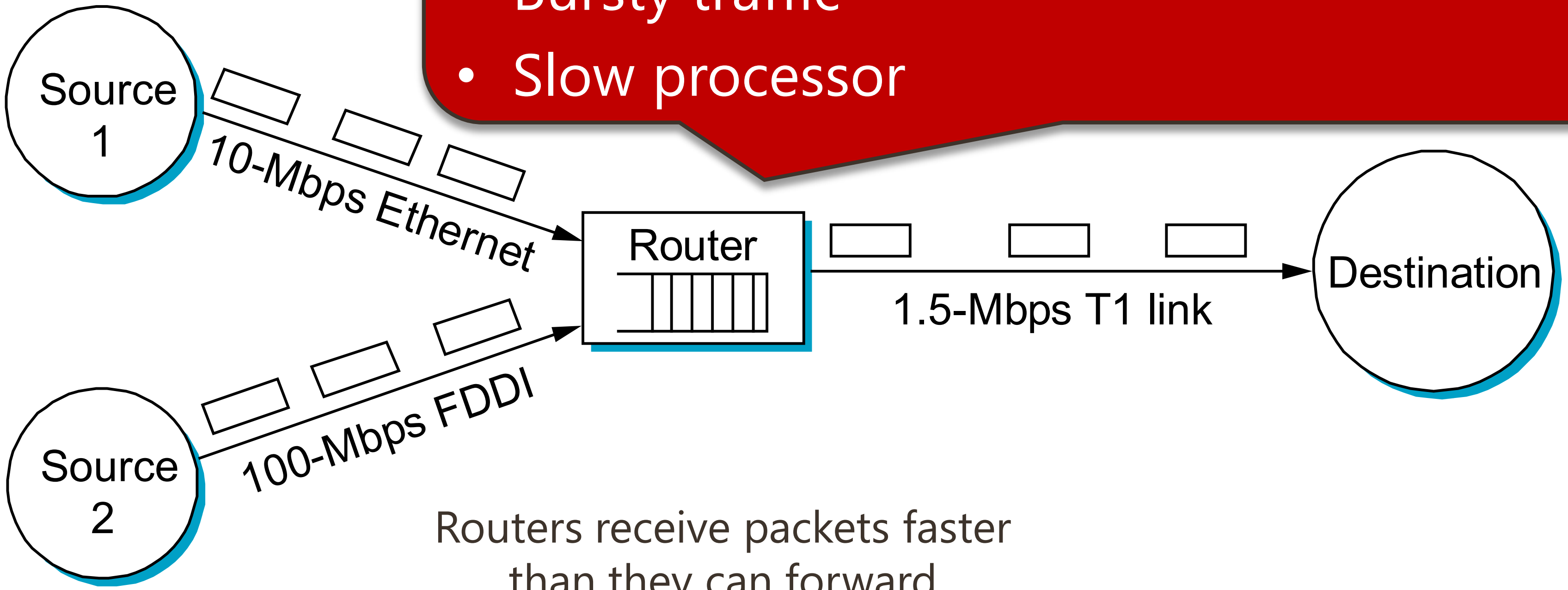
- Also used Sliding Window methods
- **Decouple Acknowledgement with available buffer**
- E.g., TCP implementation

Congestion

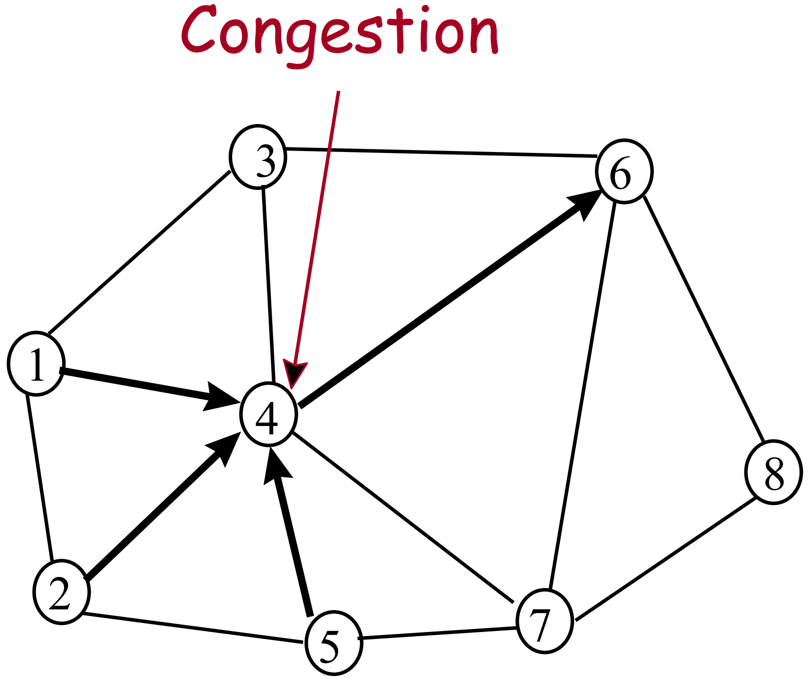
Curtin University

- One part of the subnet (e.g. one or more routers in an area) becomes overloaded, **congestion** occurs

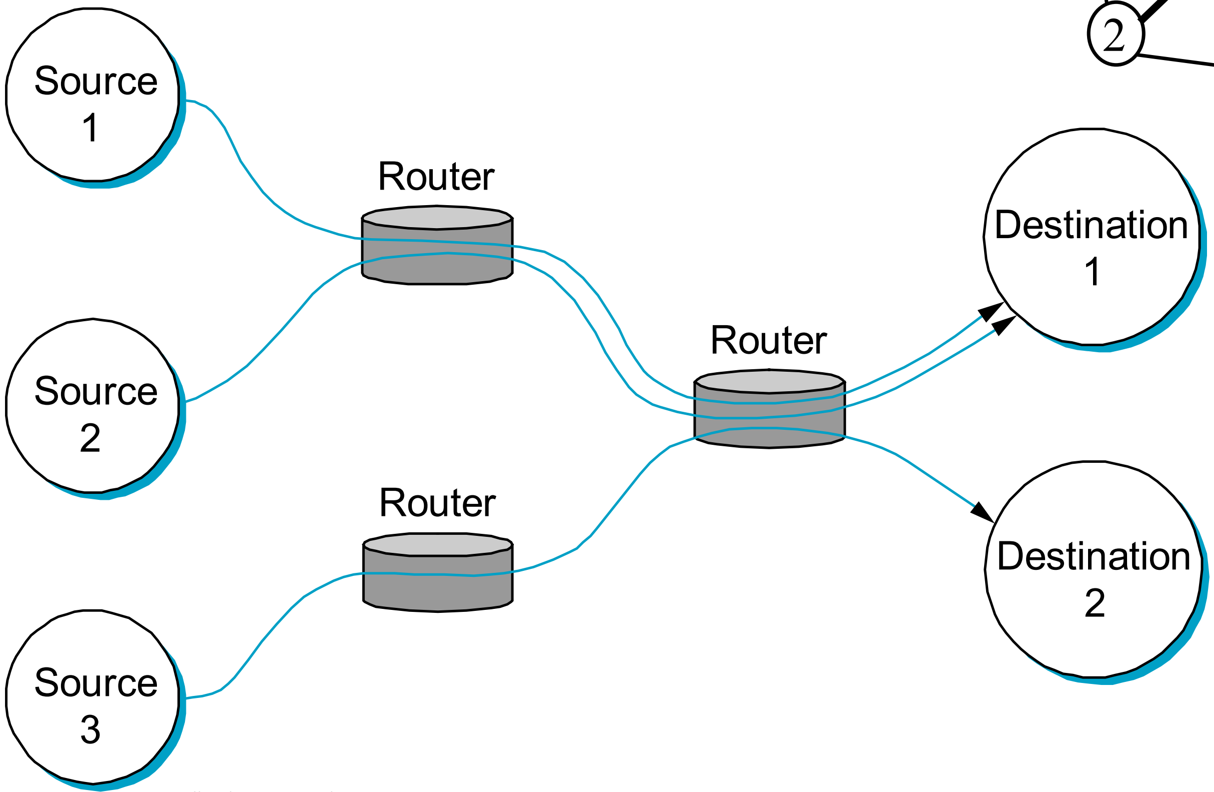
- Packet arrival rate exceeds the outgoing link capacity
- Insufficient memory to store arriving packets
- Bursty traffic
- Slow processor



Routers receive packets faster than they can forward



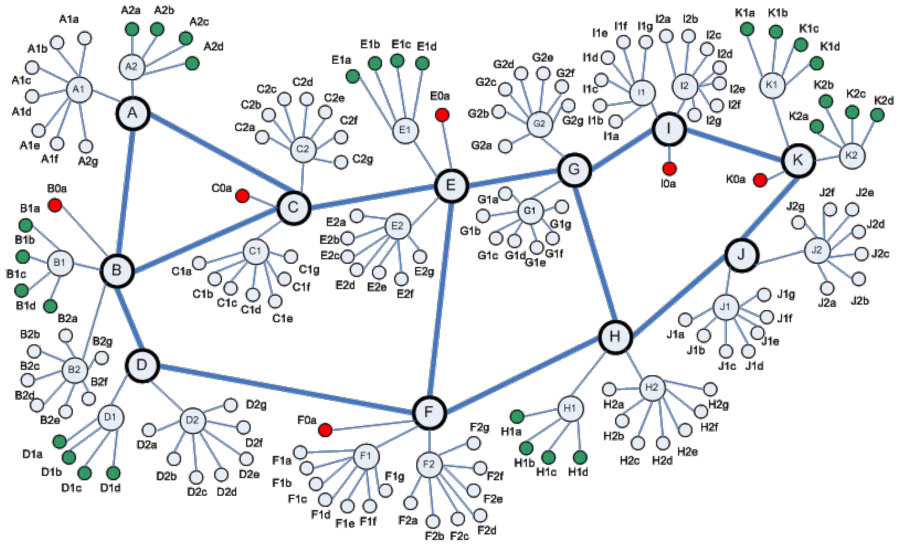
Curtin University



Congestion control can be distinguished from routing in that sometimes there is no way to *'route around'* a congested router.

Flow vs Congestion Control

▪ Congestion Control - Global Issue



Make sure the subnet is able to carry the offered traffic; **involves** all the **host**, **routers**, **store-and-forwarding** processing, etc. within the subnet

1

▪ Flow Control – Scope is **point-to-point;**

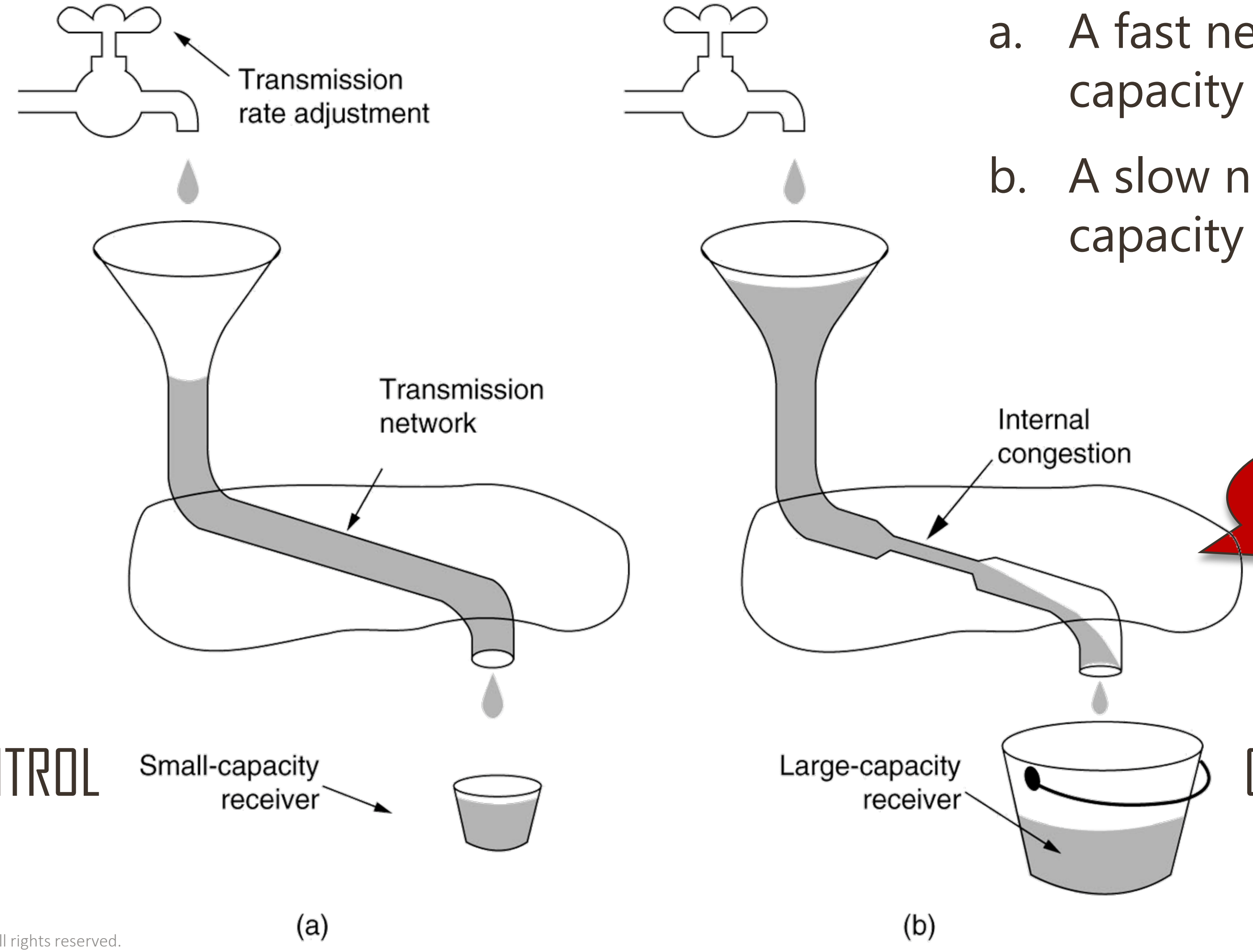
- involves just **sender** and **receiver**

Flow vs Congestion Control

Curtin University

a. A fast network feeding a low-capacity receiver

b. A slow network feeding a high-capacity receiver



FLOW CONTROL

Congestion CONTROL

reduced throughput
long response time

Congestion Control - Solutions

Curtin University

- Congestion Control is concerned with efficiently using a network at high load
- Several Solutions:

1. **Warning bit**
2. **Choke Packets**
3. **Load Shedding**

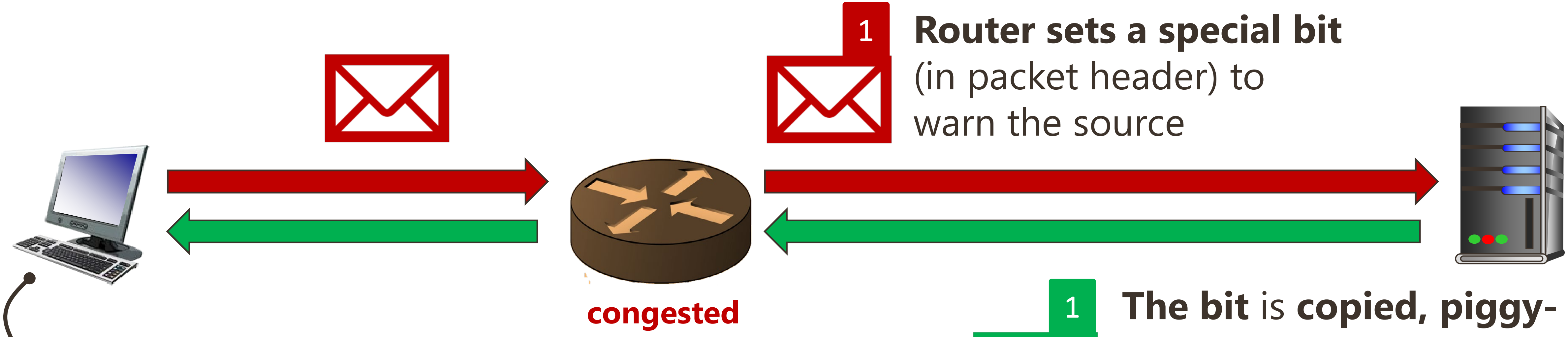
Congestion Detection & **Recovery**

4. **Random Early Discard (RED**

Congestion Avoidance

1. Warning Bit

Curtin University



Router sets a special bit (in packet header) to warn the source

The bit is **copied, piggy-backed** on the **ACK** and sent to the sender

Sender monitors the number of ACK packets it receives **with warning bit set** and adjusts its transmission rate accordingly

2. Choke Packets

Curtin University

- A **more direct way** of telling the **source to slow down**

ICMP Source Quench Packet

- A choke packet is a **control packet** generated at a congested node, transmitted to restrict traffic flow

- On receiving the choke packet,

  - Source will **reduce transmission rate** by a

2. Choke Packets - Hop-by-hop

Curtin University

- **Over long distances or at high speeds choke packets are not very effective.**

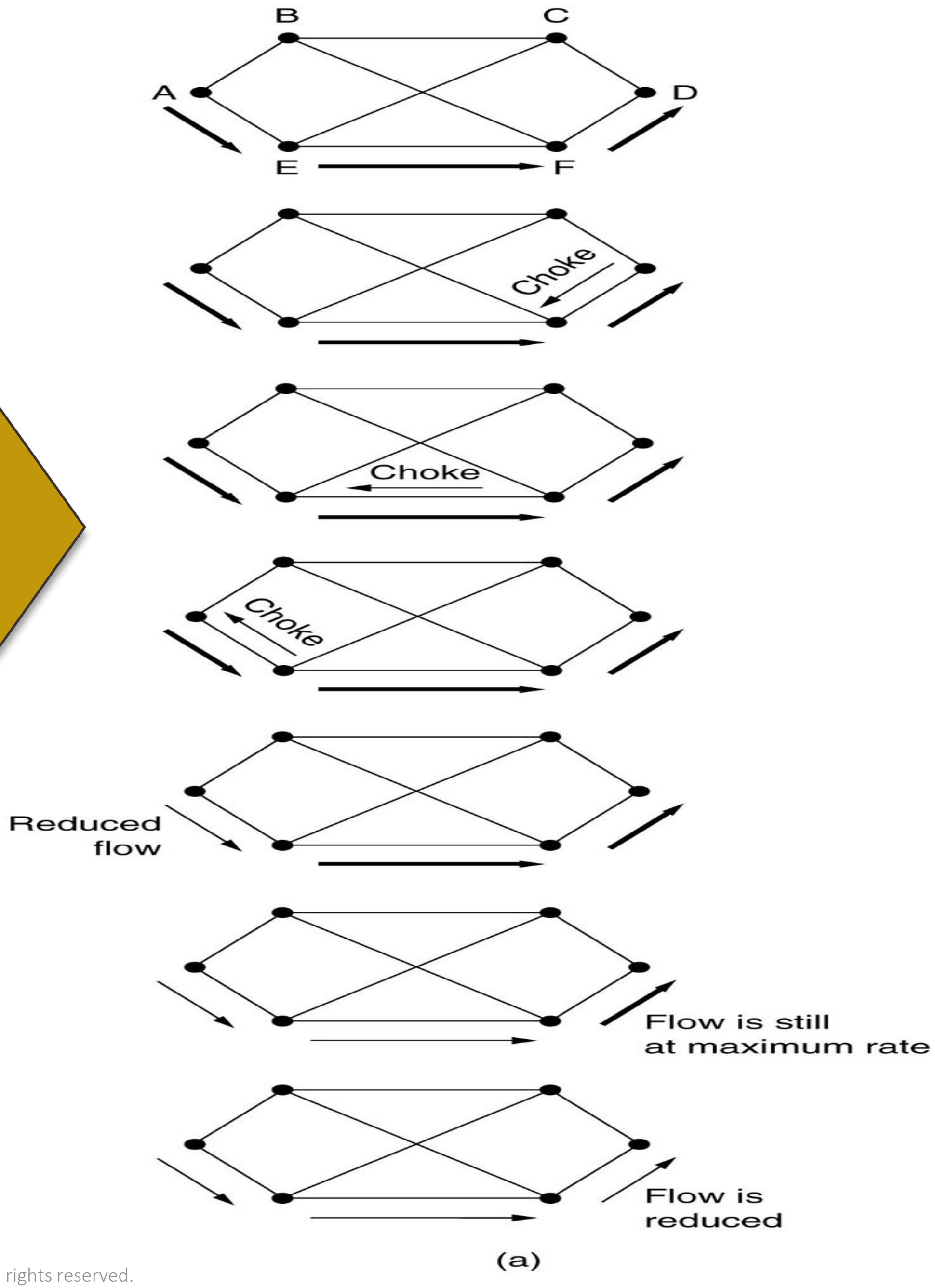
- A more efficient method: **Send the choke packets hop-by-hop**

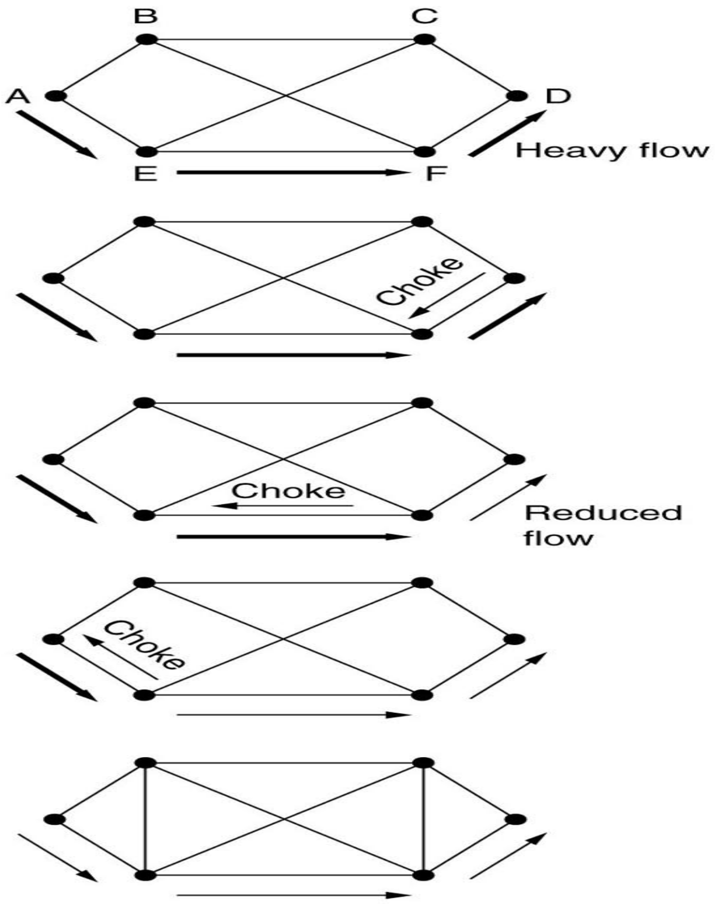
- This requires **each hop to reduce** its transmission even before the choke packet arrive at the source

Hop-by-Hop

Curtin University

Typical Choke Packet





Hop-by-Hop Choke

a. A choke packet that **affects only the source**

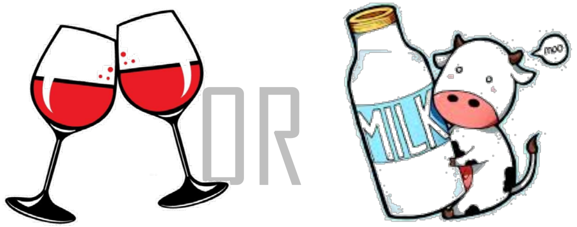
b. A choke packet that **affects each hop** it passes through (Tanenbaum)

3. Load Shedding

Curtin University

- When **buffers become full**, routers simply **discard packets**
- Which packet is chosen to be the victim depends on:

1. **Wine or Milk policy**



   - x File transfer cannot discard older packets: *will cause a gap in the received data.*
   - ✓ Real-time voice or video may throw away *old data and keep new packets.*

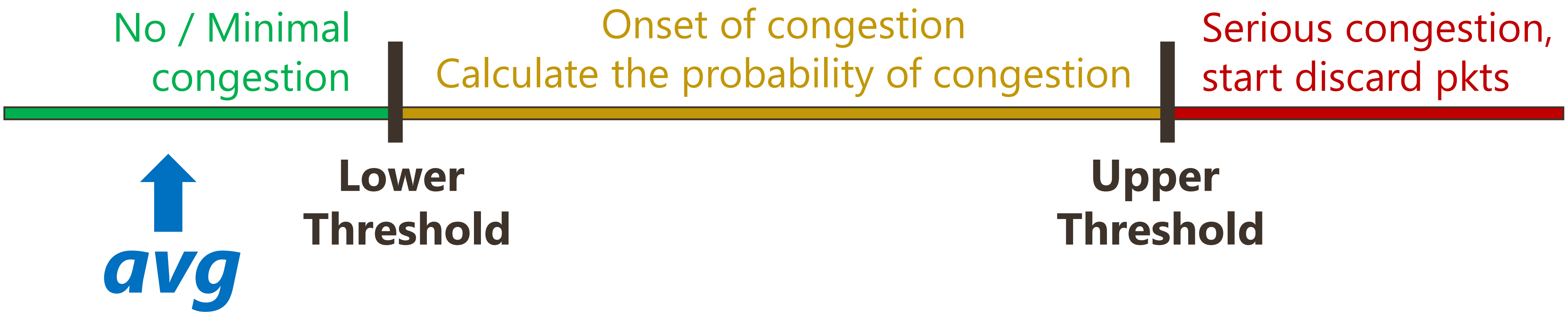
2. **Alternatively**, implement an **Intelligent Discard Policy** or get the application to mark packets with discard priority

4. Random Early Discard (RED)

Curtin University

- This is a proactive approach in which the router **discards** one or more packets ***before the buffer becomes*** *completely* ***full***

- Each time a packet arrives, the **RED algorithm** computes the average **queue length**, ***avg***



Curtin University

Transport Control Protocol (TCP)

- Fundamentals
- TCP Header
  - Flags (SYN, FIN, ACK, RST)
  - Flag (URG, PSH) – in depth
  - TCP Options
  - Window Size (Dynamic Buffer Management)
- TCP Flow Control
  - Dynamic Buffer Management

TCP

Curtin University

- The (other) **main** transport **protocol** used in the **Internet**
  - ✓ **Connection-oriented** protocol
  - ✓ RFC 793 (formal), RFC 1122 & 1323 (bug fixes)
  - ✓ Provide a reliable end-to-end communication over an unreliable internetwork

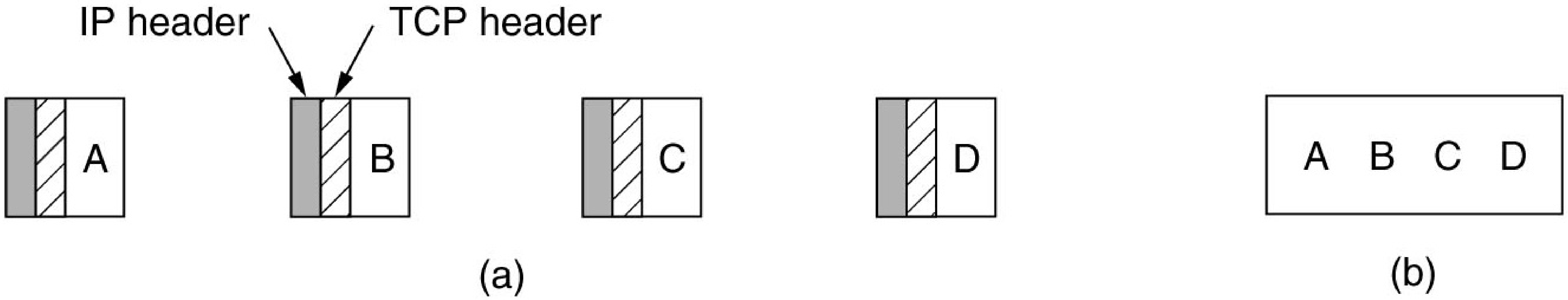
- **Connections** are:
  - ✓ **Full duplex** and point-to-point
  - ✓ **A byte stream** not a message stream

Message Boundaries are not Preserved

No support for Multicasting or Broadcasting

!

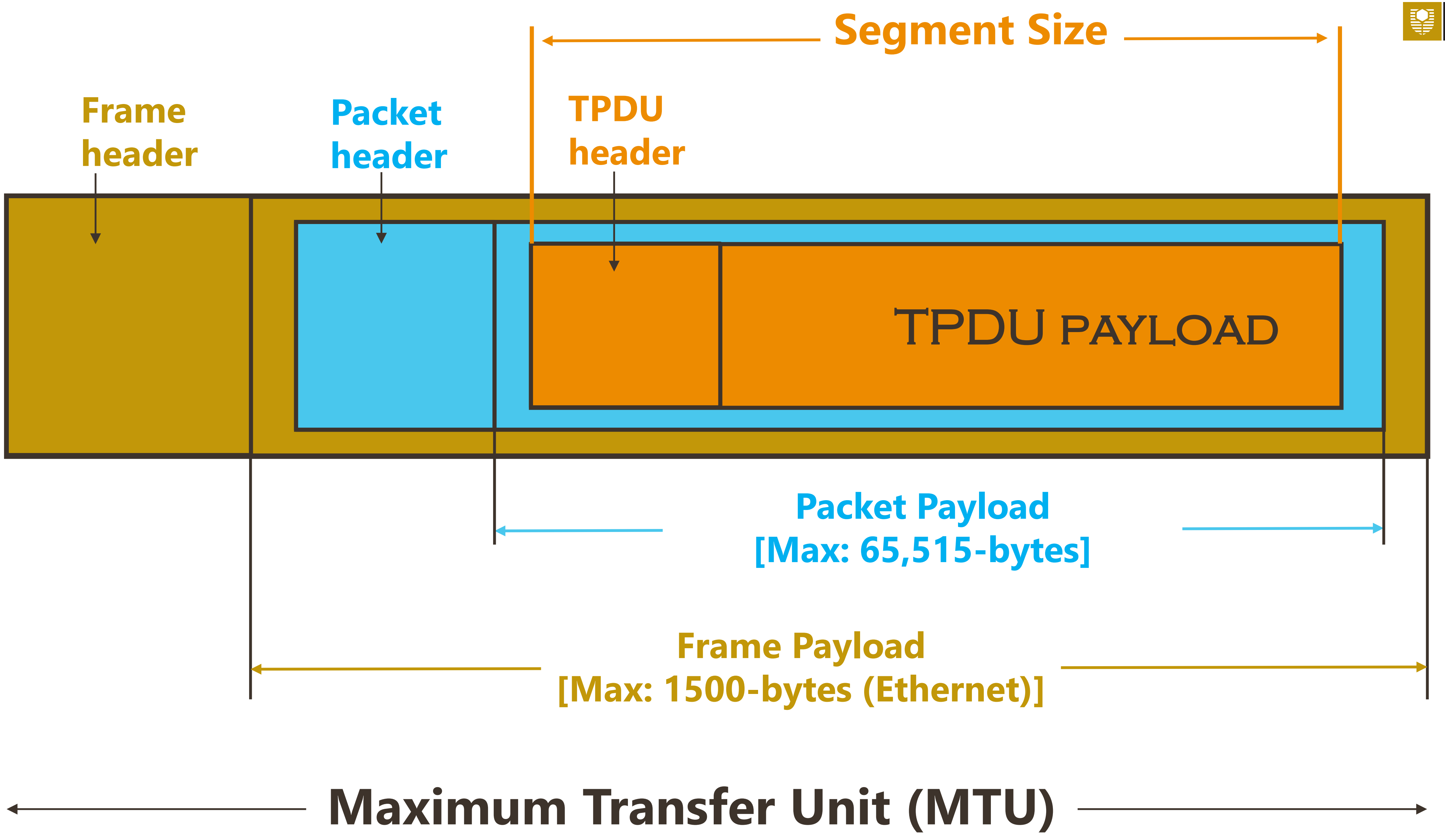
TCP: Header



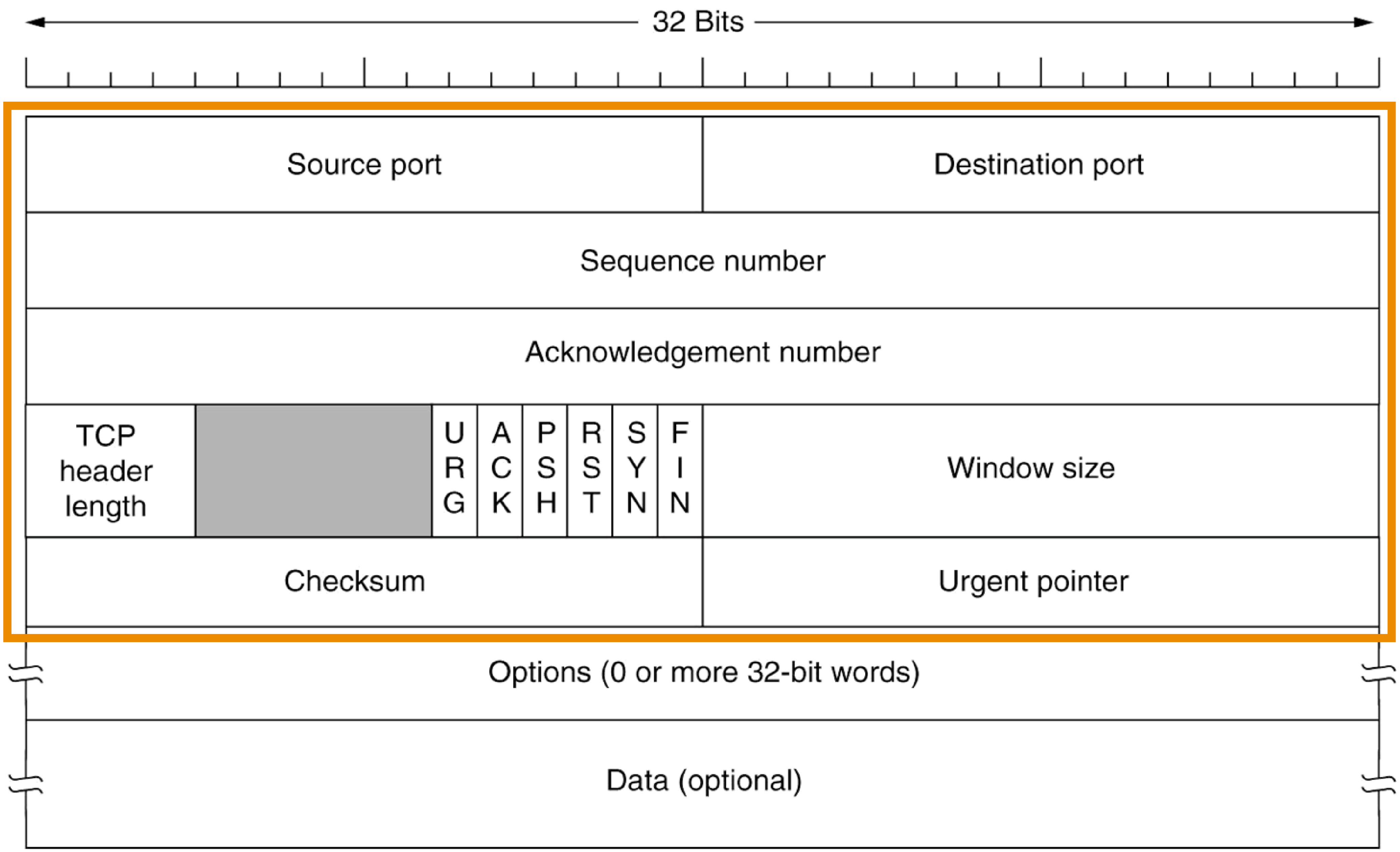
512-byte segments sent as a separated IP datagrams

2048-bytes of data received to the application in a single **READ** call

Curtin University



TCP HEADER

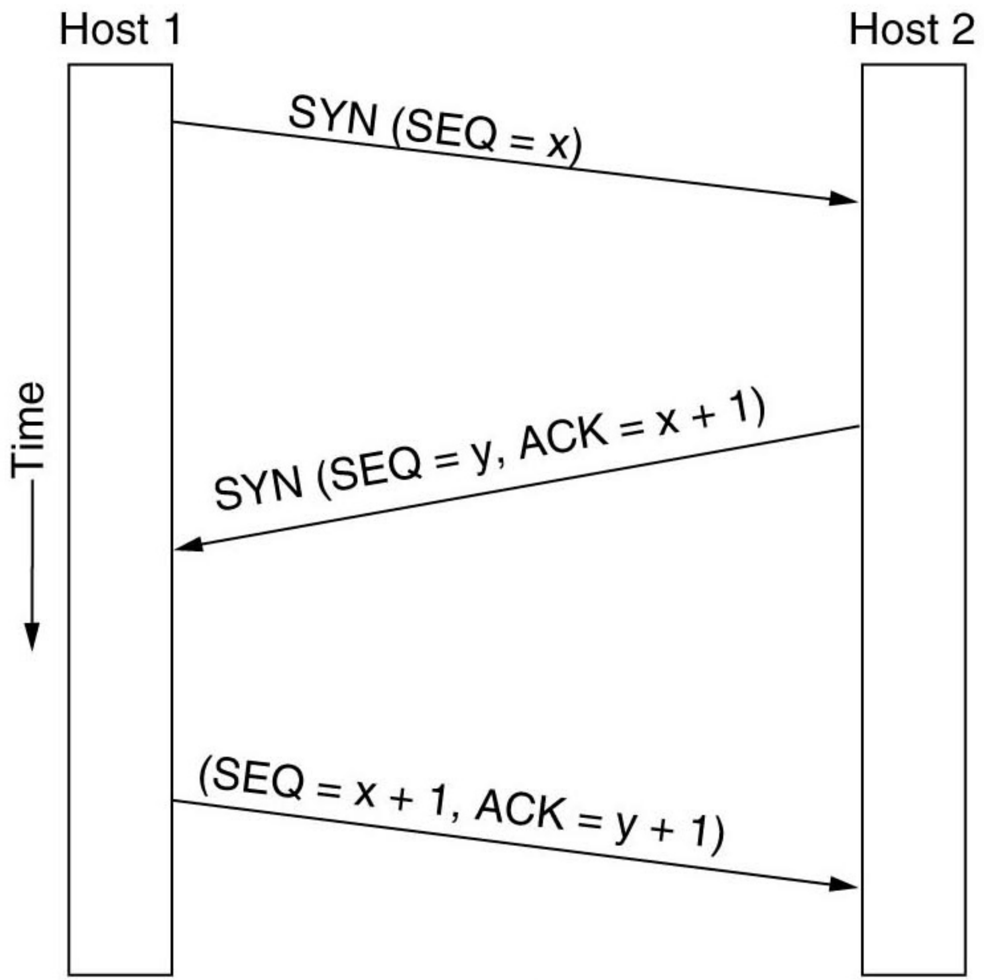


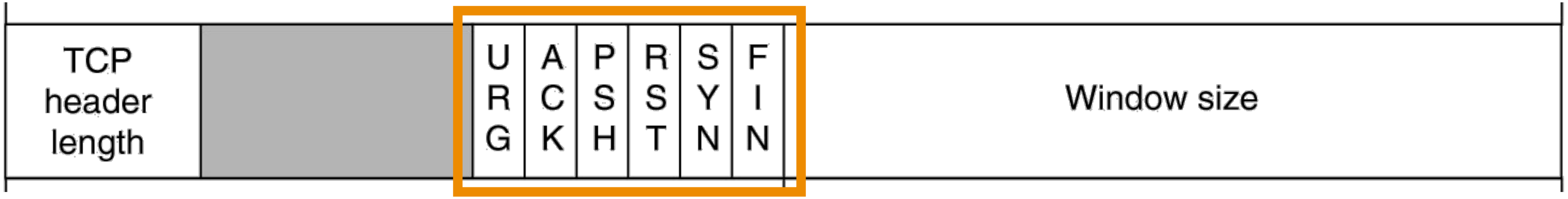
Fixed 20-bytes header

TCP: Header – cont.

Curtin University

- **Sequence Number (32-bit)**
  - sequence number of the first data octet (byte) in this segment
  - if SYN is set this field is the initial sequence number (ISN) and the first data octet is ISN+1
- **Acknowledgement Number (32-bit)**
  - sequence number of the next data octet the TCP entity expects to receive. May be piggybacked!!
  - NOTE that TCP is byte or stream oriented.





FLAGS

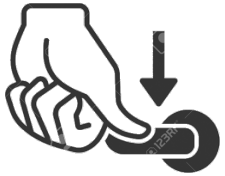
- ✓ **URG:** Urgent pointer field significant. Inform the destination TCP user that 'urgent' data is arriving.
- ✓ **ACK:** Acknowledgement field significant.
- ✓ **PSH:** Push function. A TCP user can require TCP to send (receive) all outstanding data up to and including that labelled with a **PUSH** flag.
- ✓ **RST:** Reset the connection.
- ✓ **SYN:** Synchronize the sequence numbers. Used to establish connections.
- ✓ **FIN:** No more data from sender. Used to release connections.

Curtin University

TCP header length		URG	ACK	PSH	RST	SYN	FIN	Window size
Checksum								Urgent pointer

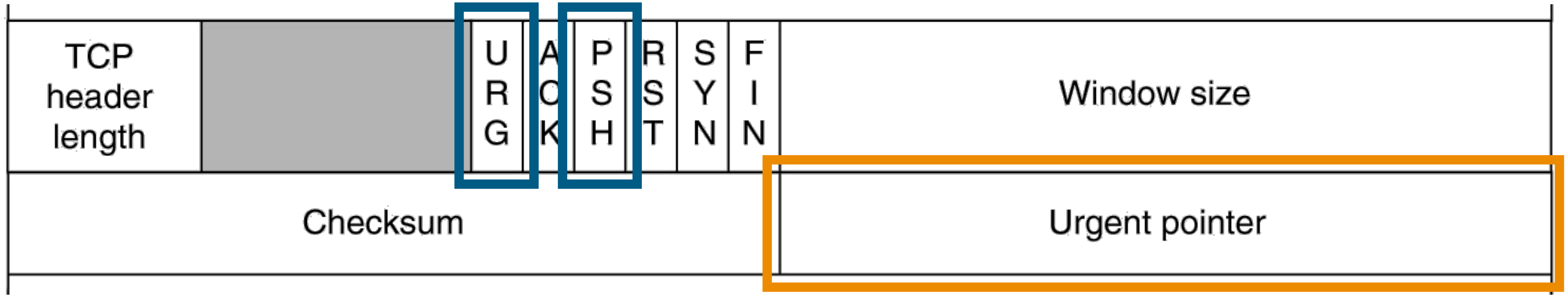
FLAGS

When application passes data to TCP, TCP may ***send*** *it* ***immediately*** *or* ***buffer it (at both sides)***
(in order to collect a larger amount to send at once)

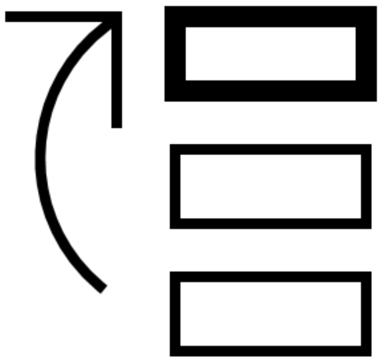
- **PUSH Flag – (used by application) ->** Force/Flush data out  ***at both sides!***
- **URGENT Flag – (used by application) ->** Cause TCP to <u>stop accumulating</u> data and <u>transmit</u> everything it has for the connection <u>immediately</u>

when urgent data is received, receiving application is **interrupted** !
(stops whatever it was doing) & read the data stream to find the urgent data

Curtin University



URG POINTER



Urgent Pointer

- ✓ **Points to the last octet** in a sequence of urgent data. Allows the receiver to know how much urgent data is coming

TCP header length		URG	ACK	PSH	RST	SYN	FIN	Window size
Checksum								Urgent pointer

Curtin University

CHECKSUM

- **Checksum:**
  - ✓ **header**
  - ✓ **data**
  - ✓ conceptual **pseudo-header**

source & destination addresses, segment length. This provides protection from mis-delivery

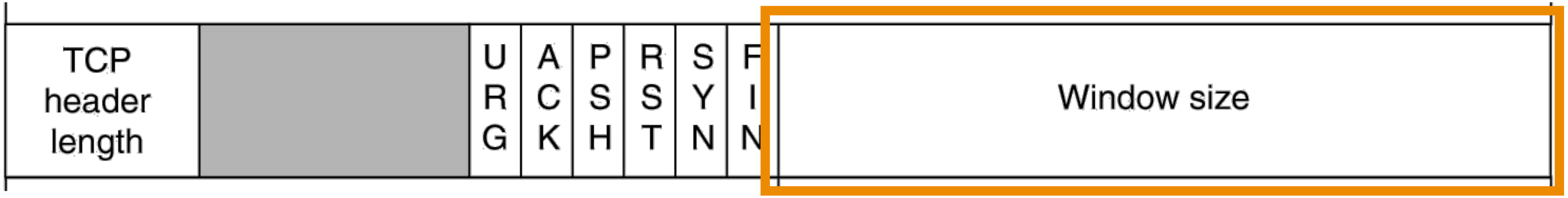
Curtin University

OPTIONS

- ✓ **Maximum Segment Size (MSS)**
- ✓ **Window Scale Factor:**

  Window is multiplied by 2F where F is the window scale factor.
- ✓ **Timestamp**

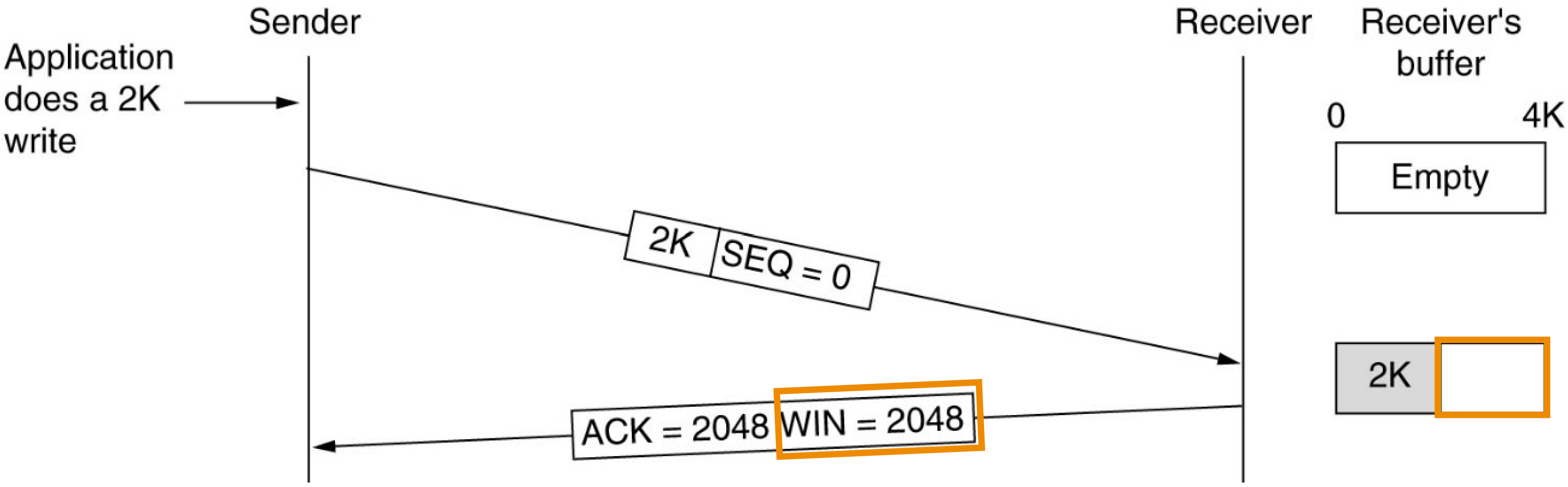
  Any outgoing packet with a timestamp will cause the ACK to carry a timestamp echo with the same value. Can be used to calculate round trip time



Curtin University

- ✓ Used in flow control
- ✓ **Variable-sized** Sliding Window

WND SIZE



Window Size

Curtin University

- **Window Management:** not tied to ACKs *(differs from data link protocols)*
- Receive entity will **advertise** window segment that it can receive & buffer
- Each entity can **alter size** of the other's sending window **dynamically** using the segment's Window field.

DYNAMIC BUFFER MANAGMENT

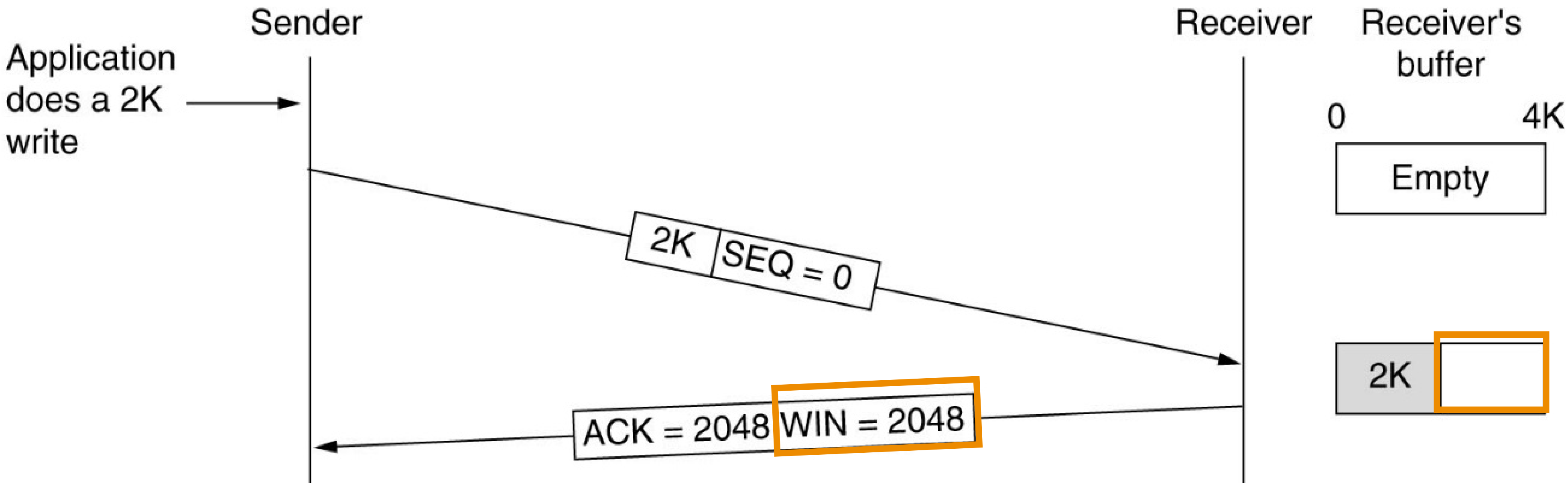
TCP: Flow Control

Curtin University

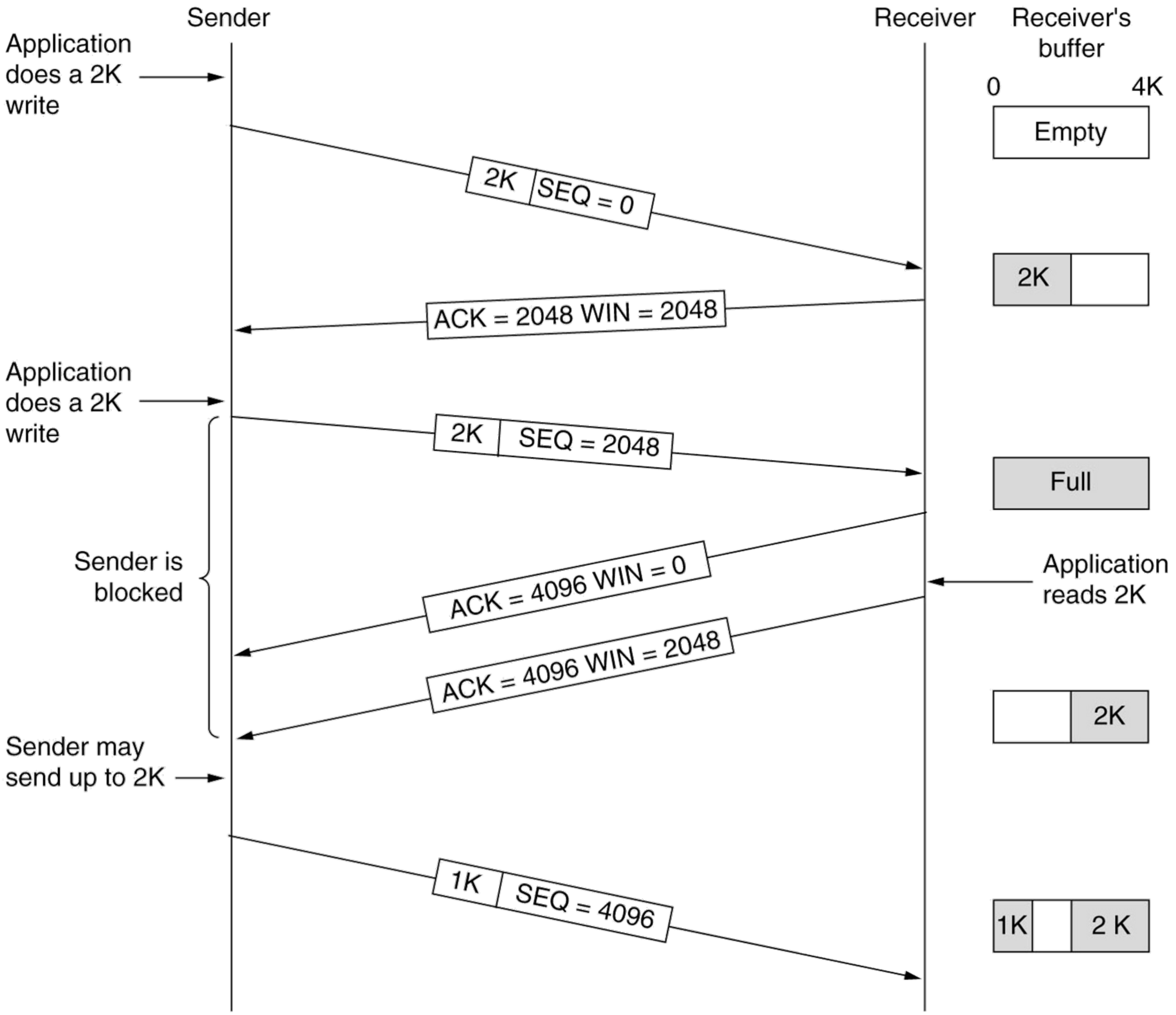
Each entity implement flow control using a **credit mechanism**, also called a **window advertisement**.

A **credit specifies** the maximum number of bytes the entity sending this segment can receive and buffer from the other entity

DYNAMIC BUFFER MANAGMENT

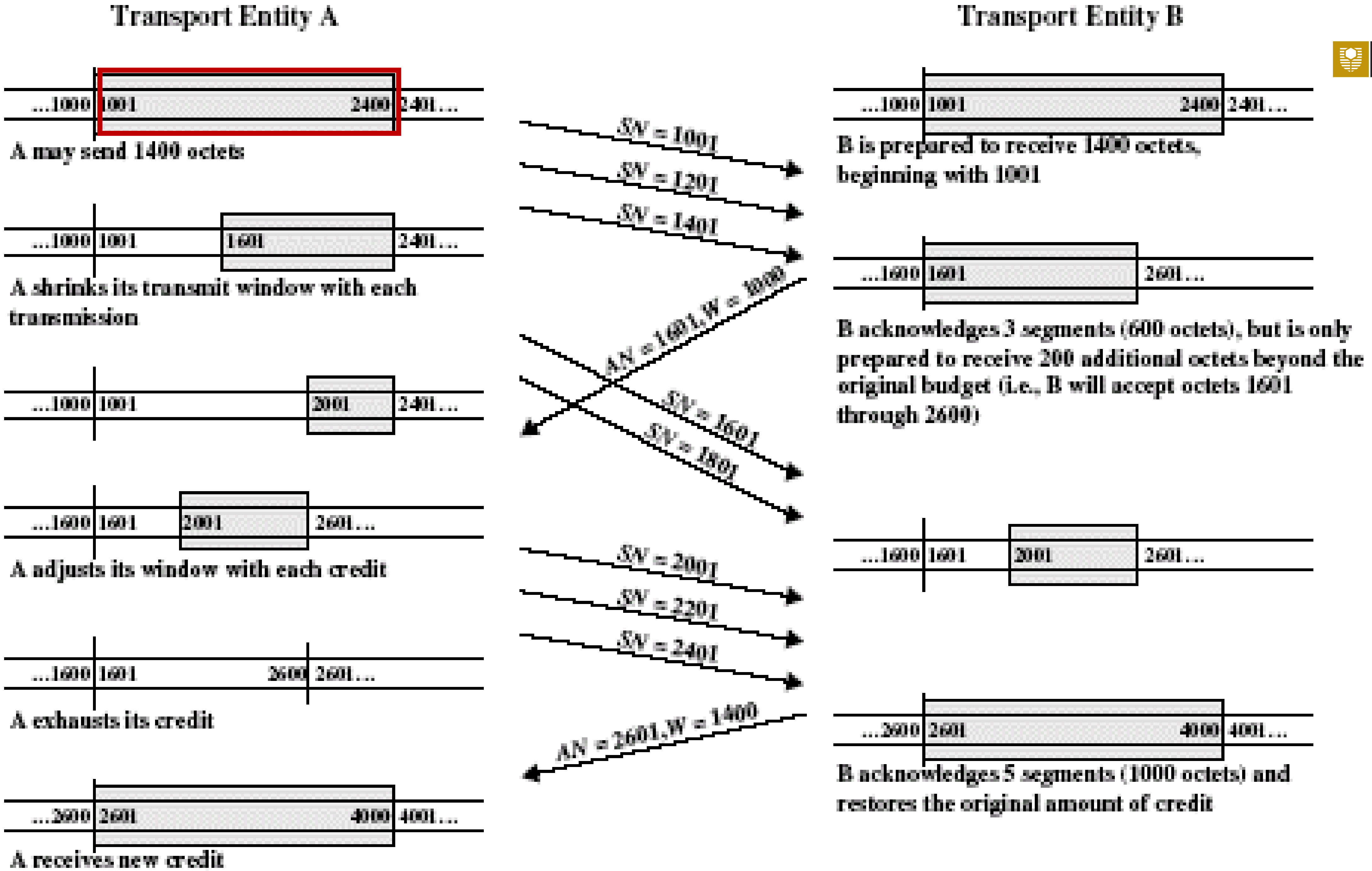


Curtin University



DYNAMIC BUFFER MANAGMENT

TCP flow control the **sequence number refers to byte sequence** instead of packet (or segment) sequences.


Transport Entity A
Transport Entity B
...1000 1001 2400 2401...
A may send 1400 octets
...1000 1001 1601 2401...
A shrinks its transmit window with each transmission
...1000 1001 2001 2401...
...1600 1601 2001 2601...
A adjusts its window with each credit
...1600 1601 2600 2601...
A exhausts its credit
...2600 2601 4000 4001...
A receives new credit
SN = 1001
SN = 1201
SN = 1401
AN = 1601, W = 1000
SN = 1601
SN = 1801
SN = 2001
SN = 2201
SN = 2401
AN = 2601, W = 1400
...1000 1001 2400 2401...
B is prepared to receive 1400 octets, beginning with 1001
...1600 1601 2601...
B acknowledges 3 segments (600 octets), but is only prepared to receive 200 additional octets beyond the original budget (i.e., B will accept octets 1601 through 2600)
...1600 1601 2001 2601...
...2600 2601 4000 4001...
B acknowledges 5 segments (1000 octets) and restores the original amount of credit

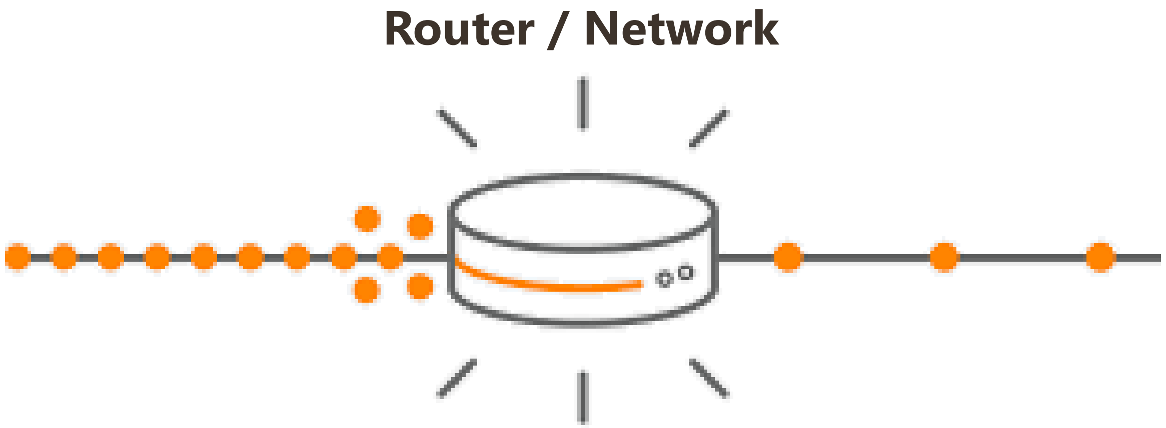
TCP Congestion Control

Curtin University

- **Network layer** *also* **tries** to manage congestion, **but difficult**

- TCP does the **heavy lifting**

"The law of conservation of packets: cannot inject new packets into the network until the old one leaves"



TCP Congestion Control

Curtin University

- **TCP dynamically manipulates the window size**
  - ✓ Detecting congestion
  - ✓ Prevent congestion (try)
  - ✓ React to congestion

- **Detecting Congestion, Difficult?**
  - Old days ☹ : **transmission error** or **packet discard** ?
  - Nowadays ☺ : **most transmission timeouts** on the Internet are due to congestion ✓

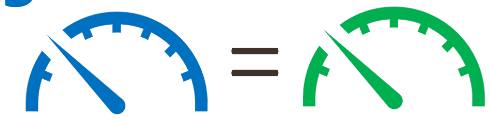
Prevent Congestion

Curtin University

"ultimately, congestion can only be controlled by limiting the total amount of data entering the internet to the amount that the internet can carry"

- **In Data Link Control Protocol**
  - ✓ Sliding window mechanism helps **to pace the sender**

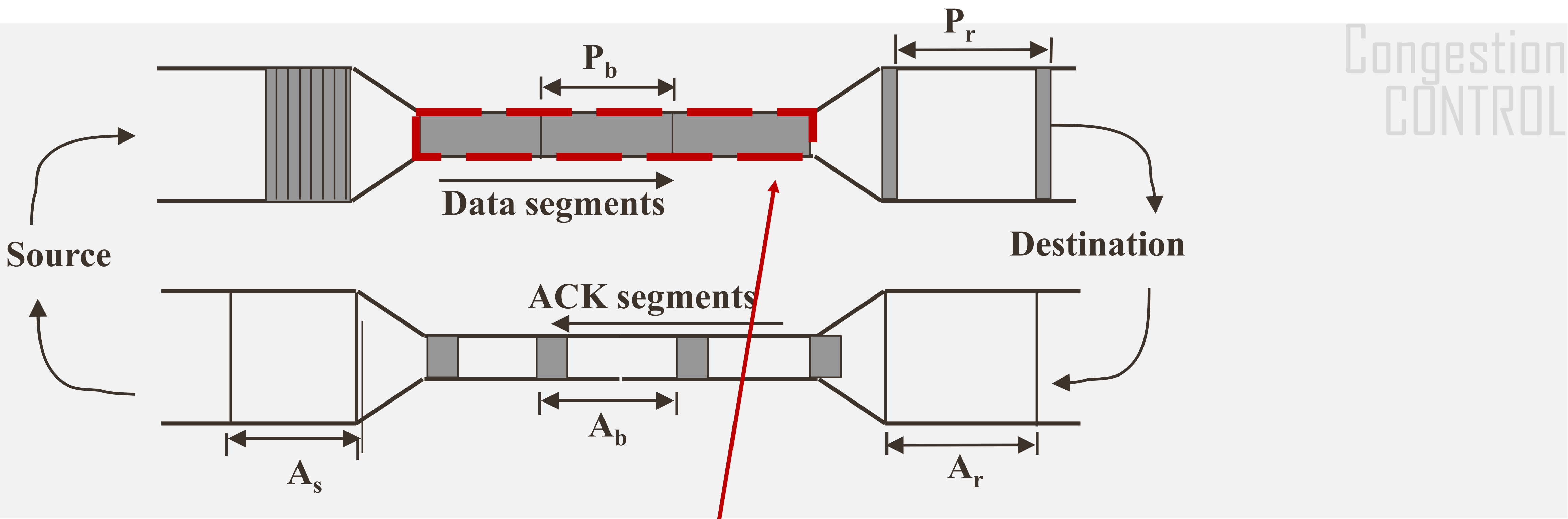
- **In TCP**
  - same **pacing effect** as data link protocol
  - **Data Sending Rate = Ack Arrival Rate**, once any initial credit is used up



Flow Determined by **Network**

Curtin University

- P_b= time of minimum segment spacing on the slowest link



Rate of ACK arrival is dependent on **round trip time**

1. bottlenecks in the **network**
2. bottlenecks at the **destination**

TCP Segment Pacing

Curtin University

- The thickness of the pipe is proportional to the data rate
- Source and destination are on high capacity networks
- **Lower speed links create bottlenecks**
- Segment represented by rectangle areas - spreads out if data rate is low
- The wider spacing is preserved at the destination even though the data rate on the final link is high, therefore the segment spacing at the receiver is: **Pr = Pb**

TCP Segment Pacing – cont.

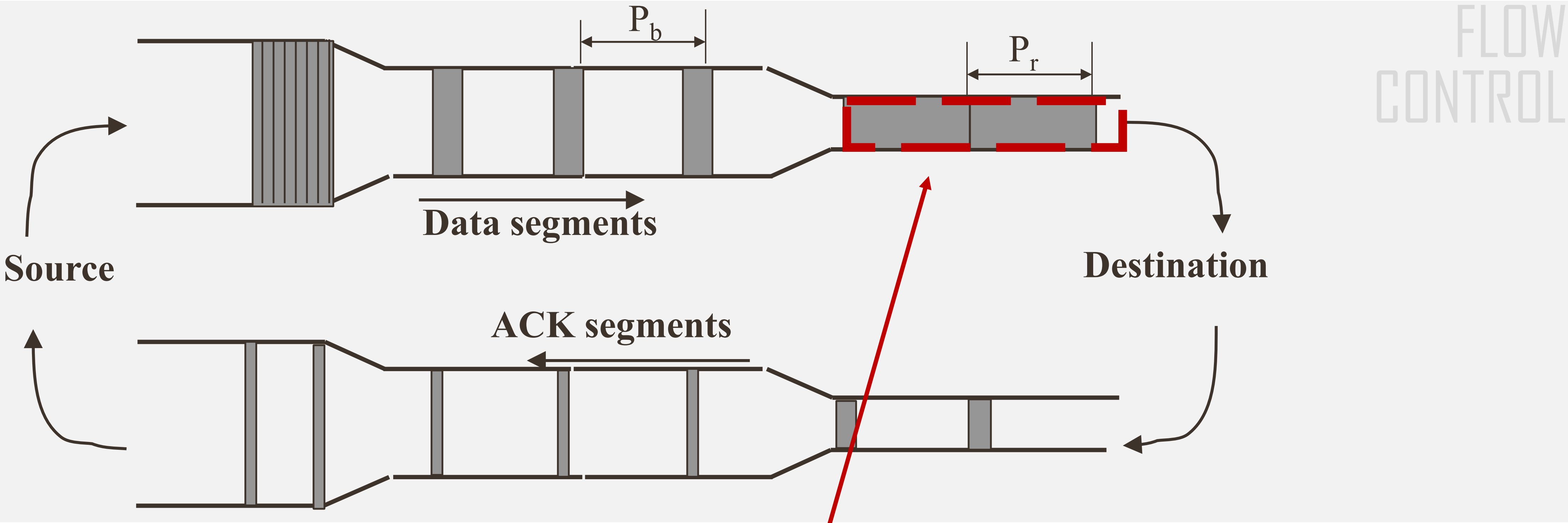
Curtin University

- In **steady state**,
  - ✓ sender's segment rate = arrival rate of ACKs
  - ✓ sender's rate is determined by the slowest link on the path
- Sending TCP entity automatically **senses and regulates flow**
  - ✓ self-clocking
- Self-clocking works equally well with the bottleneck at receiver
  - ✓ ACKs are sent out at a rate equal to the absorption capacity of the destination

Flow Determined by **Destination system**

Curtin University

- P_b= time of minimum segment spacing on the slowest link



Rate of ACK arrival is dependent on **round trip time**

1. bottlenecks in the **network**
2. bottlenecks at the **destination**

TCP Congestion - Solutions

1. **Segment Pacing Effect** – ***discussed before***
2. **Slow Start**
3. **Dynamic Window Sizing on Congestion**

2. Slow Start

Curtin University

```
awnd = min{credit, cwnd}
```

- **Each sender maintains two windows**
  - **`awnd`** - Window the receiver has granted
  - **`cwnd`** - Congestion window

    each window reflects the number of bytes the sender may transmit

  - **`credit`** - amount of unused credit granted in the most recent ACK in segments

2. Slow Start – cont.

Curtin University

- **After connection is established**
  - sender initializes **cwnd** to the size of the maximum segment (normally **cwnd=1**) and sends one maximum segment
  - if segment is ACKed before timeout, **cwnd** is increase to two maximum segment size and two segments are sent
- **As each segments is ACKed,**
  - the **cwnd** is increased by one maximum segments size

cwnd grows exponentially until either a receiver's window is reached, or timeout occur

- **The idea:**
  - if bursts of size **n** bytes work fine but a burst of **n+n** bytes gives timeout, the **cwnd** is set to n bytes to avoid congestion

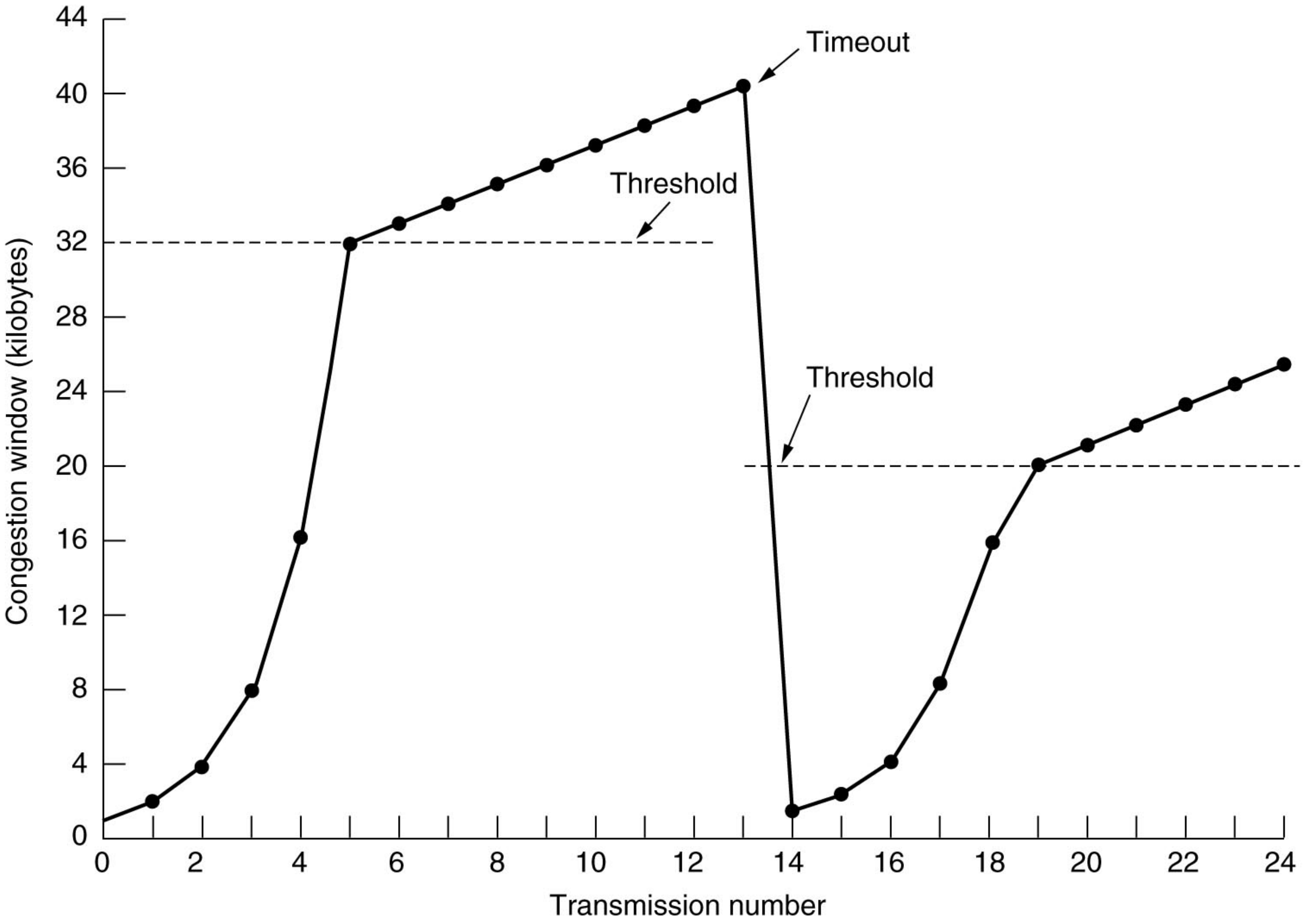
3. Dynamic Window Sizing on Congestion

Curtin University

- An internet congestion control algorithm
- if **congestion causes** a timeout, **cut back flow**, **ramp up slowly**
- **when timeout occurs**
  - `ssthresh` = slow start threshold
  - set `ssthresh` = `cwnd`/2
  - set `cwnd`=1 and **performs slow-start until** `cwnd` = `ssthresh`
  - from then on **linear grow**, increase **cwnd by 1** (segment) **after every ACK** until the receiver's window is reached or timeout occur

University

Dynamic Window Sizing



TCP Timer Management

- TCP uses multiple timer (at least conceptually) to do its work.
- Retransmission timer
  - How long should the timeout interval be?
- Determine the Round-Trip Time (***RTT***), time between sending a segment and receiving its ACK, is tricky.
- Even if ***RTT*** is known, deciding the timeout interval is also difficult
  - if timeout is set too short (say T1) - unnecessary retransmission
  - if timeout is set too long (say T2) – performance suffer due to long retransmission delay

TCP Timer Management – cont.

Curtin University

- Highly dynamic algorithm that constantly adjust the timeout interval, based on continuous measurements of the network performance.

- Jacobson's algorithm (1988) – RTT variance estimation:
  - ✓ generally **used by TCP**
  - ✓ for each connection, determine best retransmission timer (RTO) by using an estimate of RTT which includes an estimate of the variance
  - ✓ use the *mean deviation* (not standard deviation) and *exponential averaging*

TCP Timer Management – cont.

Curtin University

- for each connection, TCP entity maintains a variable *SRTT*, that is the best current estimate of *RTT* to the destination.

- if an ACK gets back before timeout, TCP measures *RTT* for the ACK, and updates *SRTT*:

$$SRTT(k+1) = (1-\alpha)\times SRTT(k) + \alpha\times RTT(k+1)$$

where α is a smoothing factor that determines how much weight is given to the old value. Typically $\alpha = 1/8$

- even with good value of *SRTT*, selecting the retransmission timeout is still difficult

Curtin University

- Normally, TCP uses $\beta \times SRTT$
  - Trick is selecting β
  - it was initially constant value $\beta = 2$ (inflexible)

- Jacobson proposed making β roughly proportional to the *mean deviation* (not standard deviation) of the acknowledgment arrival time probability density function

- keeping track of another smooth variable, D

$$D = \alpha \times D + (1-\alpha)|SRTT-RTT|$$

where α may or may not be the same value used to smooth $SRTT$

- most TCP implementations now uses Jacobson's algorithm to set the timeout interval.

$$RTO = SRTT + 4 \times D$$

Karn’s Algorithm

Curtin University

What happen during timeout retransmission

Which RTT samples should be used as input to Jacobson’s algorithm

- **Karn’s proposed a simple fix to Jacobson algorithm**
  - Do not update **SRTT** on any segments that have been retransmitted
  - Timeout is doubled on each failure until the segments get through the first time

Exponential Backoff

Curtin University

- What retransmission timer (RTO) value should be used on a retransmitted segment?

> TCP source increases its RTO value each time the same segment is retransmitted - backoff process

- After the first retransmission of a segment
  - wait for a longer time before performing a second retransmission

$$RTO = q \times RTO$$

  - RTO grows exponentially after each backoff (common value of q is 2)

Curtin University

TCP
Implementation Policies

- Send Policy
  - Silly Window Syndrome
- Deliver Policy
- Accept Policy
- Retransmit Policy
- Acknowledge Policy

TCP: Implementation Policies

Curtin University

- The standard provide a precise specification of the protocol to be used between TCP entities

- Policies defined for the protocol
  1. **Send policy**
  2. **Deliver policy**
  3. **Accept policy**
  4. **Retransmit policy**
  5. **Acknowledge policy**

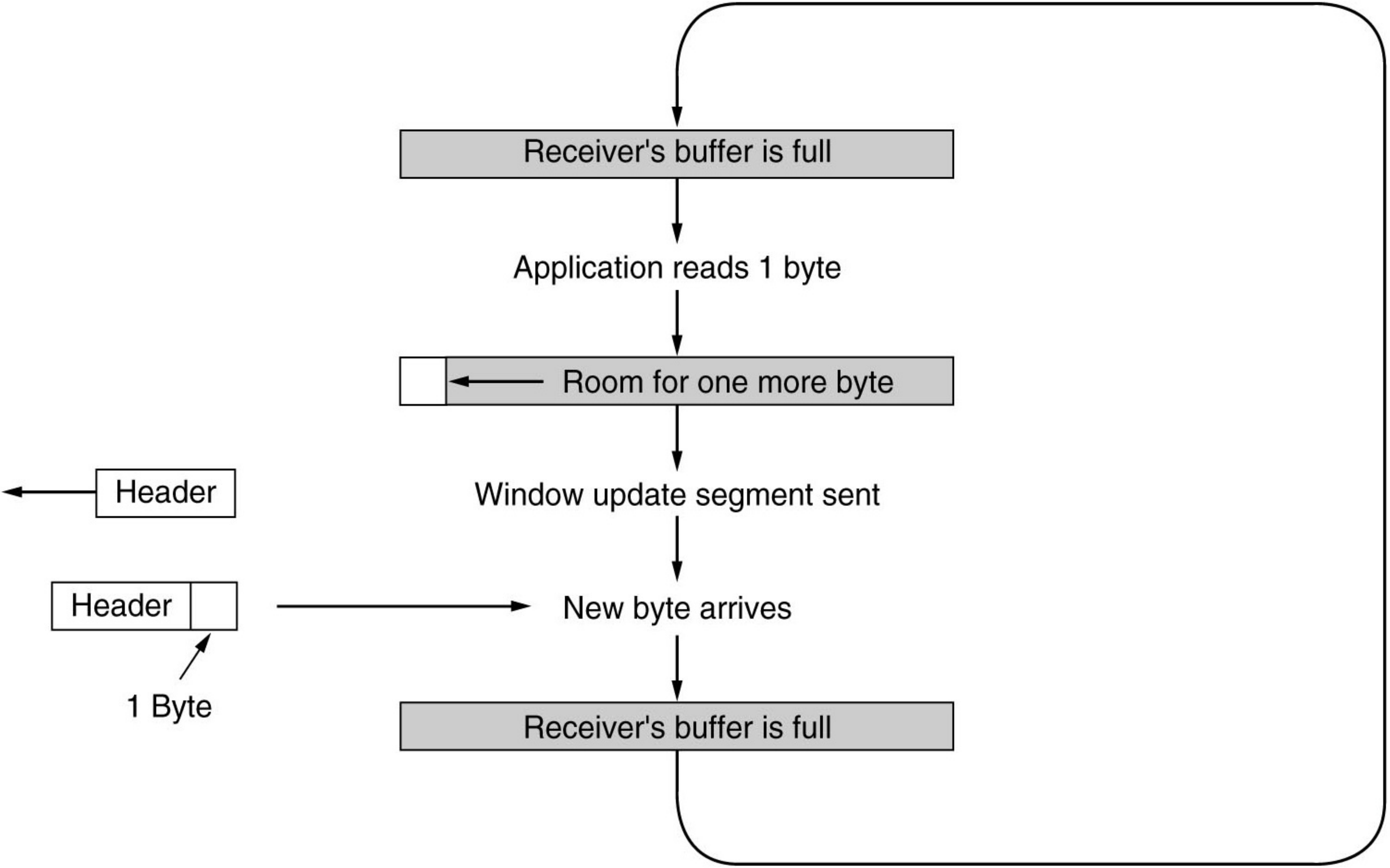
1. Send Policy

Curtin University

In the **absence of PUSH** data and a closed transmission window, a sending TCP entity is free to **transmit data at its own convenience**

- depend on performance considerations
  - ✓ if **infrequent and large** *(buffer @ sender)*
    - low overheads
    - slow response
  - ✓ if **frequent and small** *(buffer @ receiver)*
    - quick response
    - high overheads
    - silly window syndrome

Silly Window Syndrome



2. Deliver Policy

Curtin University

Too frequent delivery means **too many OS interrupts**

- Arriving data are stored in deliver buffer

✓ **if PUSH flag is set**

Data along with any other data in the deliver buffer are submitted to the destination application in a *RECEIVE* command.

✓ **if PUSH flag is not set**

TCP may wait, *e.g. to avoid excess interrupts*

✓ **if URG flag is set**

The receiving application is signaled that urgent data is present

Curtin University

3. Accept Policy

- **In Order**
  - ✓ **Discard** out of order segments
- **In Window**
  - ✓ **Accept** all segments within the receive window
  - ✓ Complex acceptance test and sophisticated storage

out of order segments are accepted

4. Retransmit Policy

Curtin University

TCP maintains a **queue of segments** that have been sent but **not ACKed**

- **First Only** suited for **in-window accept policy**
  - ✓ one retransmission timer for the entire queue
  - ✓ if an ACK is received, the segment/s removed and timer reset
  - ✓ if timer expires, first segment in the queue is retransmitted

Selective-Repeat ARQ

- **Batch** suited for **in-order accept policy**
  - ✓ same as above, except when timer expires, retransmit all segments in the queue

Go-Back-N ARQ

- **Individual** suited for **in-window accept policy**
  - ✓ one timer for each segment

Selective-Repeat ARQ

5. Acknowledge Policy

Curtin University

- **Immediate**
- **Cumulative**
  - ✓ wait for an outbound segment, piggyback the ACK
  - ✓ Timer to avoid long delay

Curtin University

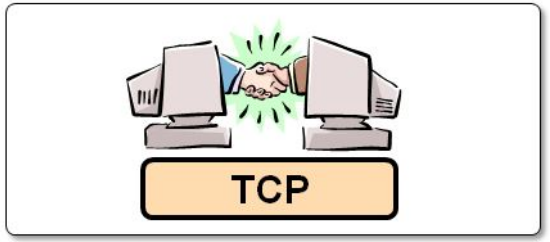
User Datagram Protocol (UDP)

- Fundamentals
- Applications
  - DNS
  - RPC

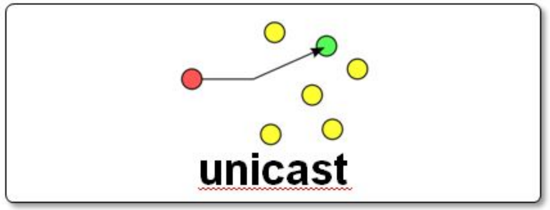
Protocols: UDP [RFC 768]

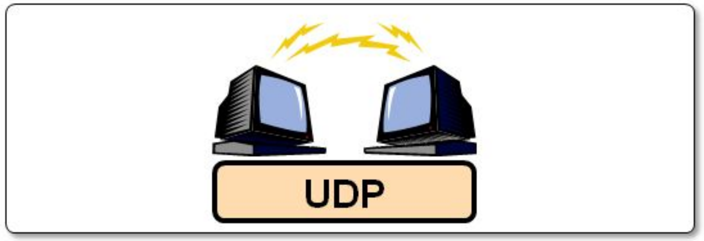
Curtin University

- **TCP: Reliable ordered delivery** of packets
  - Error detection, retransmissions and acknowledgements.
  - TCP is strictly used for **point to point**
  - TCP segments the data before sending to Network layer – Stream oriented
- **UDP: Unreliable delivery** of packets
  - Connectionless protocol
  - No retransmissions, acknowledgements
  - UDP does not segment the data – message oriented

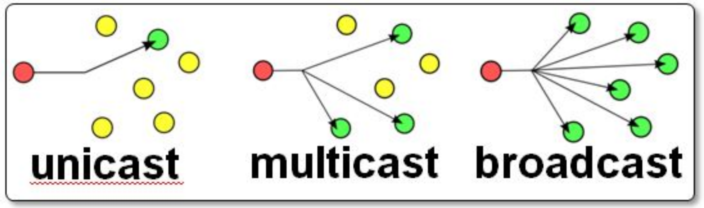


- **Slower but reliable transfers**
- **Typical applications:**
  - Email
  - Web browsing





- **Fast but non-guaranteed transfers ("best effort")**
- **Typical applications:**
  - VoIP
  - Music streaming



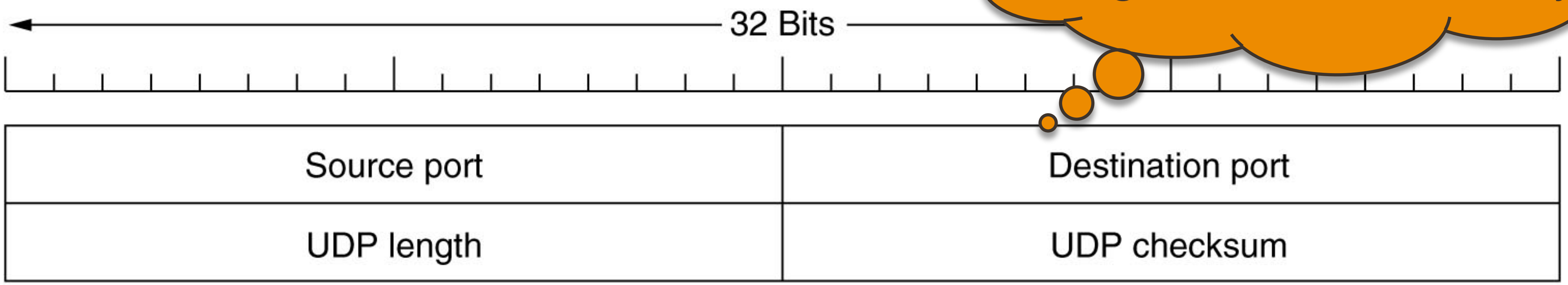
No flow control, error control, or retransmission upon receipt of bad segments

Why UDP?

Curtin University

- Lower overhead enables faster transmissions
- **Unicast**, **Multicast**, **Broadcast**
- UDP is especially useful in client-server situation
  - ✓ DNS (Domain Name System) uses UDP

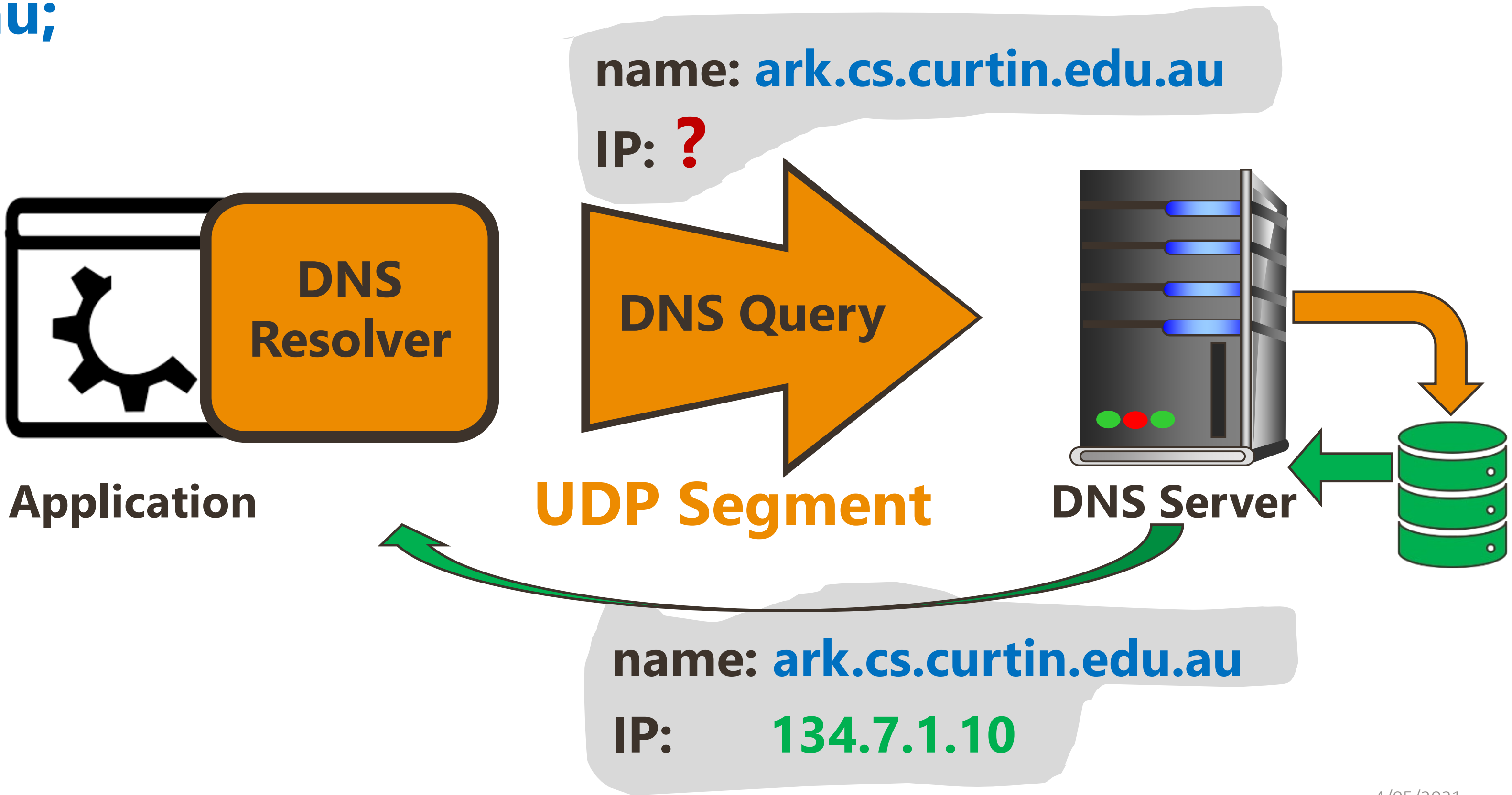
TCP protocol requires more information and overhead to guarantee data delivery



UDP: DNS (Domain Name System)

Curtin University

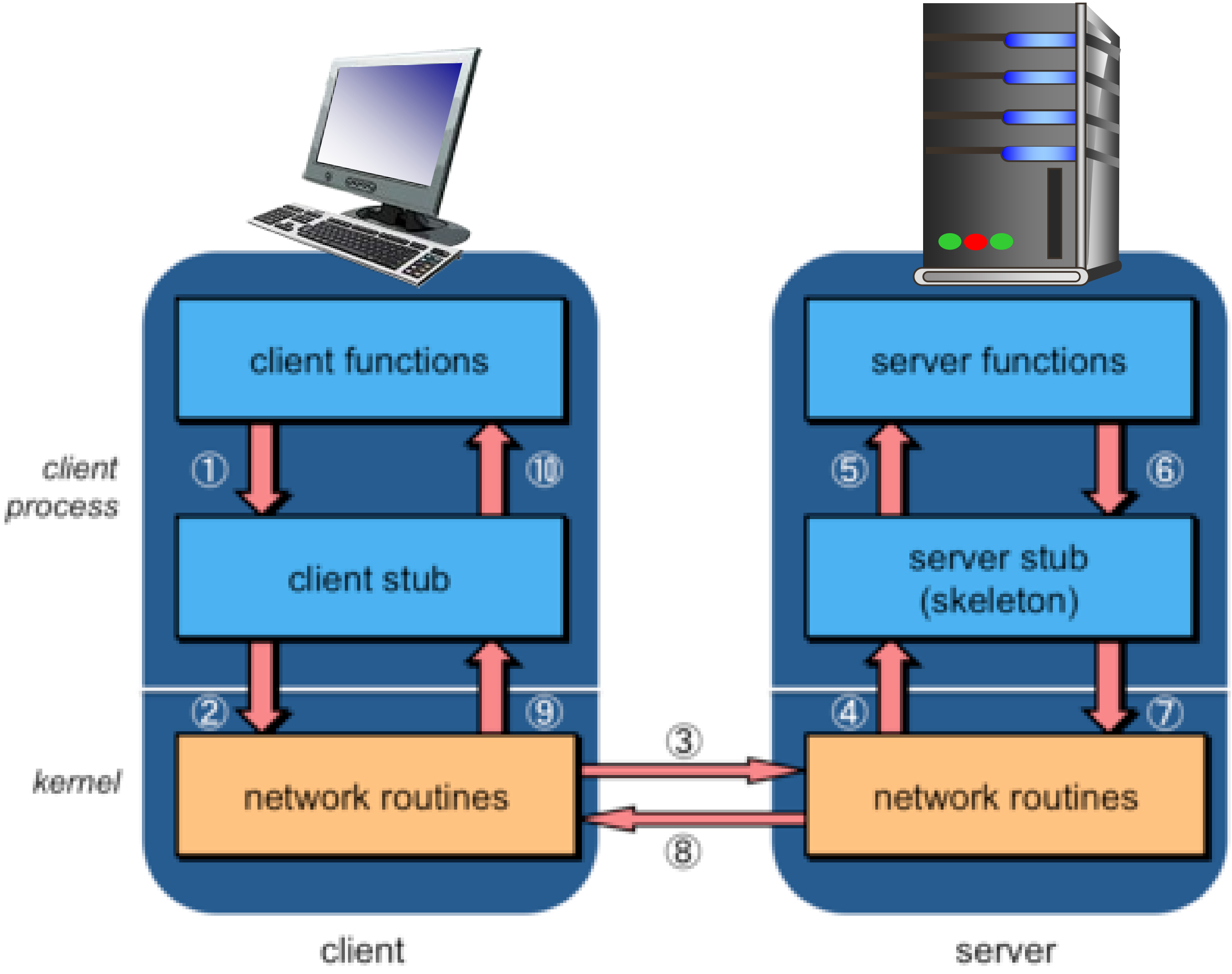
- How can the **IP address of the remote host** be found?
- Hierarchical, symbolic addresses are used
  - e.g. **ark.cs.curtin.edu.au;**
  - curtin.edu.au



Remote Procedure Call (RPC) - 1984

Curtin University

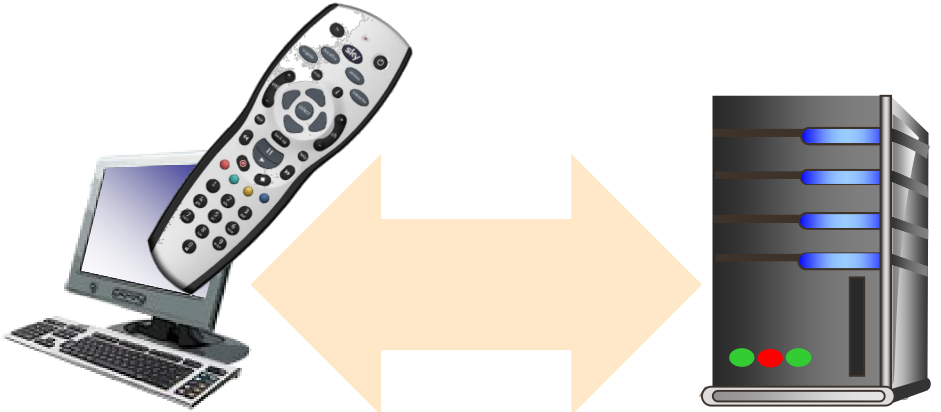
- **Allows program to call procedures located on remote hosts**
  - Information is transported from the caller to the callee (remote host) in the parameters and can come back in the procedure result
  - **Message passing is invisible** to the programmer



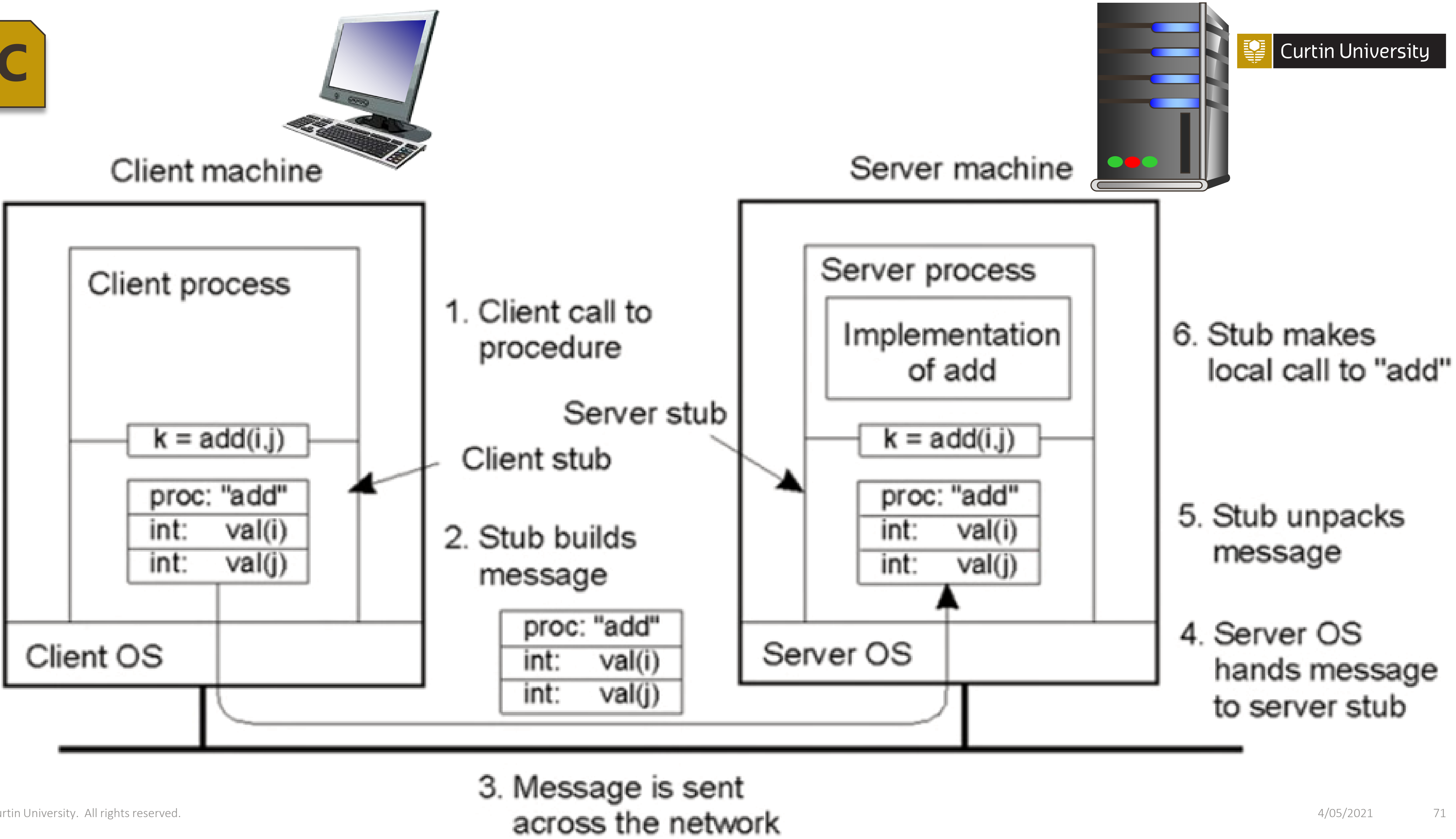
RPC – cont.

Curtin University

- **Client Stub**
  - **a small library procedure** that represents the server procedure in the client's address space that the Client program must be bound with.
- **Server Stub**
  - **a procedure call that represents the client procedure call** in the server's address space that the Server program must be bound with.
- **UDP is commonly used for RPC**



RPC



Client machine
Server machine
Client process
Server process
Implementation of add
k = add(i,j)
proc: "add"
int: val(i)
int: val(j)
1. Client call to procedure
Server stub
Client stub
2. Stub builds message
6. Stub makes local call to "add"
5. Stub unpacks message
4. Server OS hands message to server stub
Client OS
Server OS
3. Message is sent across the network

Curtin University

- **Flow Control and Congestion Control**
  - Flow Control
    - Transport Layer
    - Data Link Layer
  - Congestion Control
    - Warning-bit
    - Choke-packets
    - Load-shedding
    - RED

- **TCP**
  - TCP Header
    - Flags (SYN, FIN, ACK, RST)
    - Flag (URG, PSH) – *in depth*
    - TCP Options
    - Window Size (Dynamic Buffer Management)
  - TCP Flow Control
    - Dynamic Buffer Management
  - TCP Congestion Control
    - TCP Segment Pacing Effect
    - Slow Start
    - Dynamic Window Sizing
  - TCP Timer Management

- **TCP - Implementation Policies**
  - Send Policy
    - Silly Window Syndrome
  - Deliver Policy
  - Accept Policy
  - Retransmit Policy
  - Acknowledge Policy

- **UDP**
  - Fundamentals
  - Applications
    - DNS
    - RPC

Curtin University

THANK YOU

Make tomorrow better.