L4 – practical examples

MŰEGYETEM 1782

Budapest University of Technology and Economics

Department of Telecommunications and Media Informatics

Ports

- A connection
  - 5-tuple
- Setup a connection
  - UDP – no setup
  - TCP
    - 3 way handshake
    - Why it is needed?

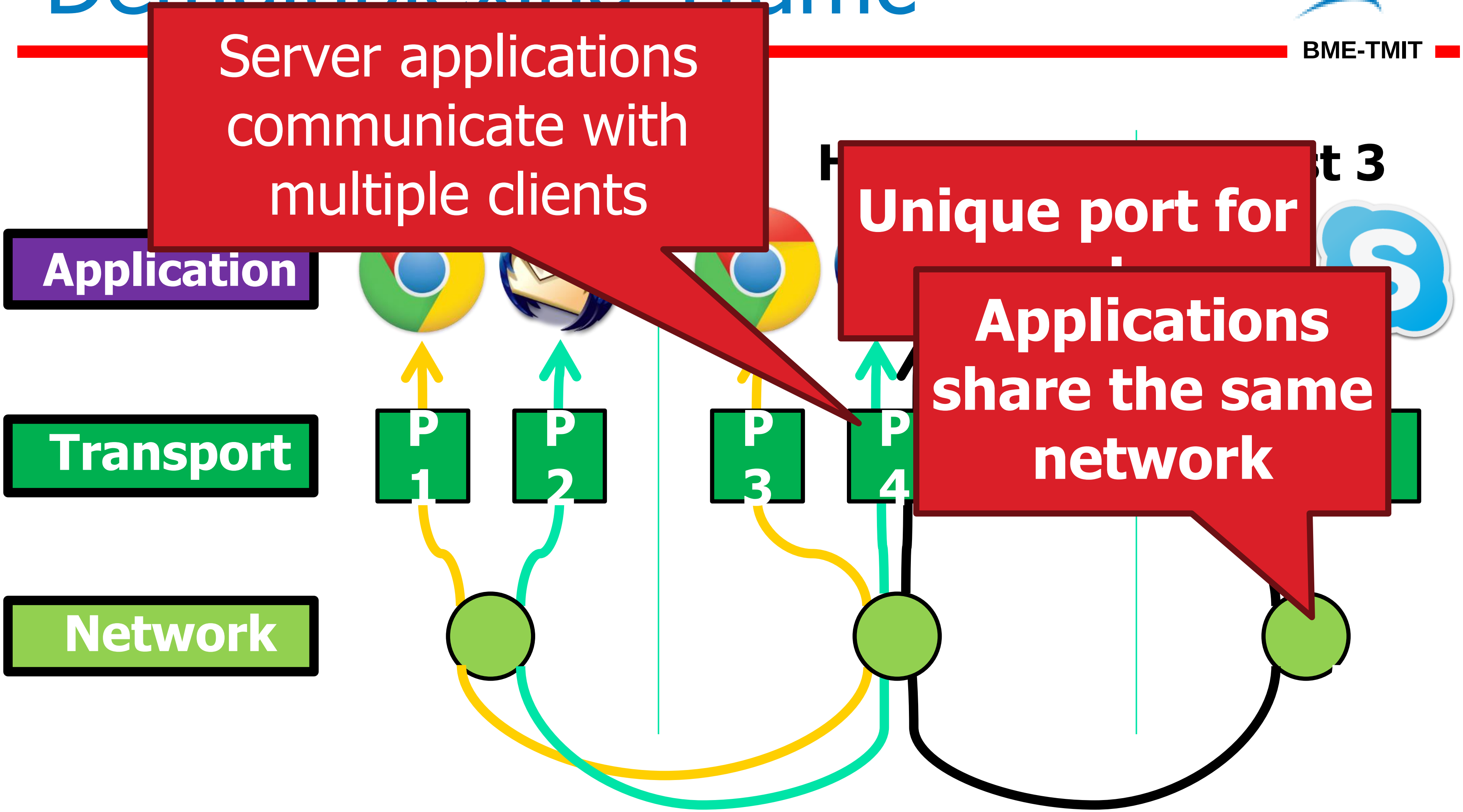
Ports on a computer

- Netstat command

TCP	127.0.0.1:1906	localhost:1907	ESTABLISHED
TCP	192.168.1.147:53699	13.77.87.52:https	ESTABLISHED
TCP	192.168.1.147:53703	91.190.216.57:12350	ESTABLISHED
TCP	192.168.1.147:53737	64.4.23.152:40008	ESTABLISHED
TCP	192.168.1.147:53759	108.177.96.188:5228	ESTABLISHED
TCP	192.168.1.147:53772	40.77.226.192:https	ESTABLISHED
TCP	192.168.1.147:54512	a104-96-129-73:https	CLOSE_WAIT
TCP	192.168.1.147:54513	a104-96-129-73:https	CLOSE_WAIT
TCP	192.168.1.147:54514	a104-96-129-73:https	CLOSE_WAIT

Demultiplexing Traffic

BME-TMIT



Endpoints identified by *<src_ip, src_port, dest_ip, dest_port>*

UDP

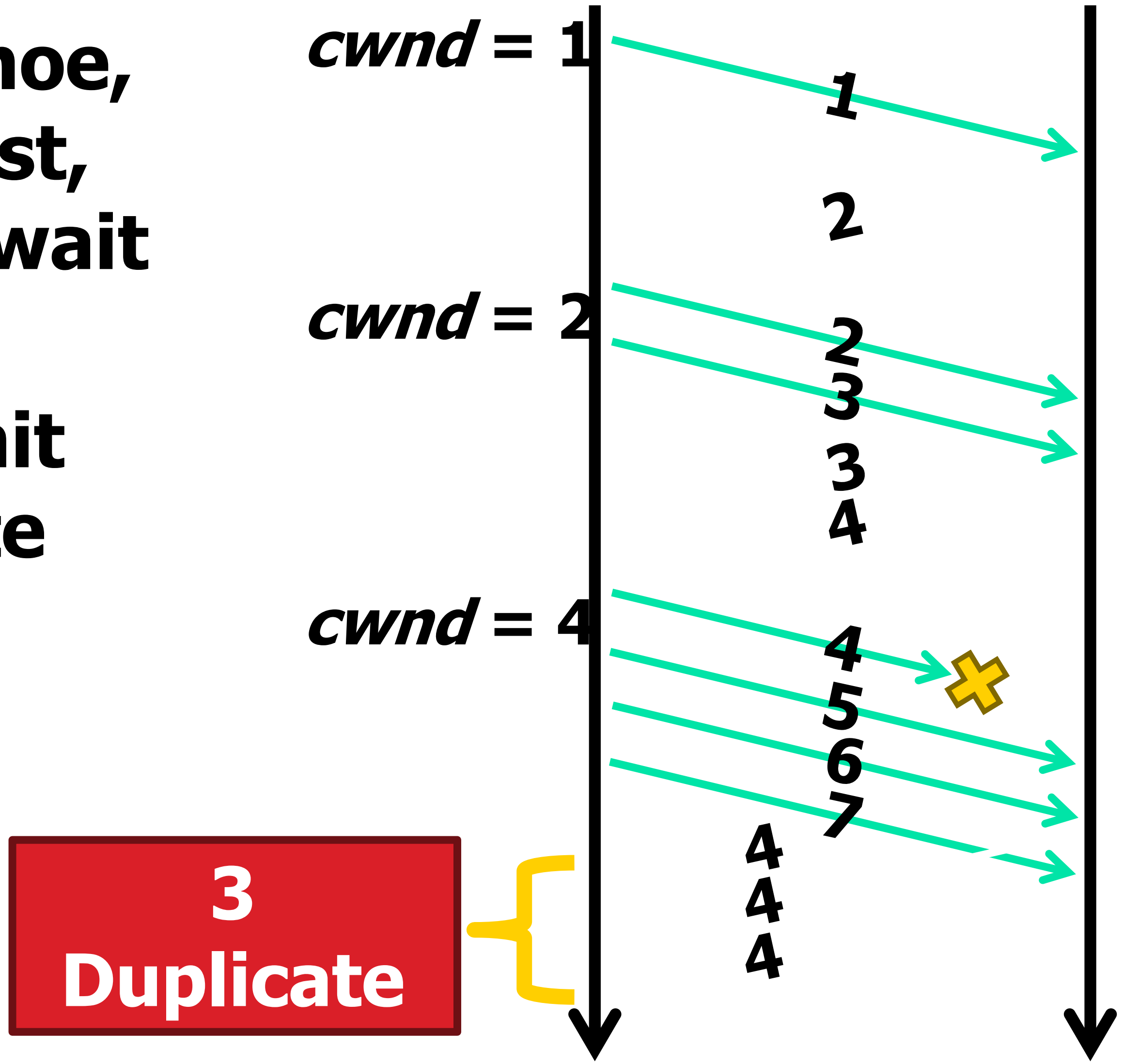
- Data segments
- Error handling
  - ICMP: port unreachable
  - Loss: no feedback
- Delay, bandwidth
  - Multicast!

The Evolution of TCP

- TCP Tahoe
  - Initial version
- The TCP was developed in 1974!
  - Today there are many versions of TCP
- A widely spread initial version: TCP Reno
  - Tahoe, plus…
  - Fast retransmit
  - Fast recovery

TCP Reno: Fast Retransmit

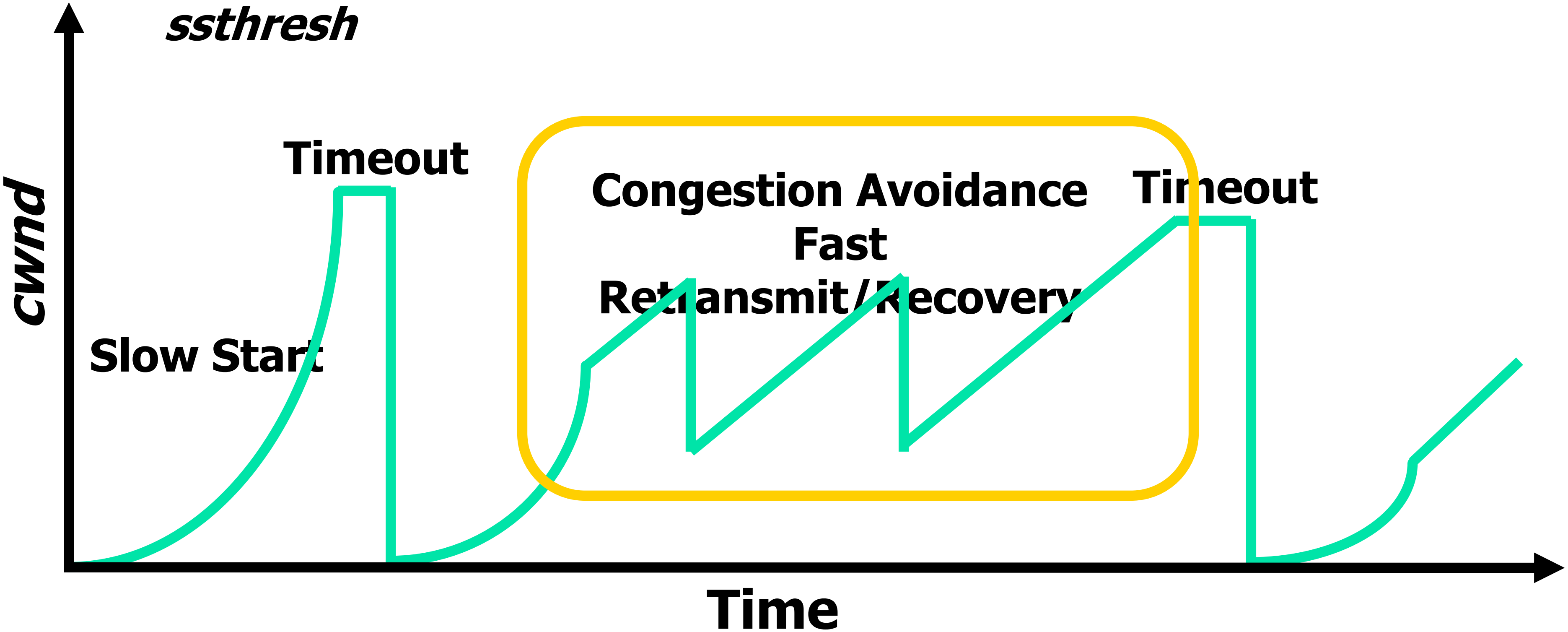
- **Problem: in Tahoe, if segment is lost, there is a long wait until the RTO**
- **Reno: retransmit after 3 duplicate ACKs**



TCP Reno: Fast Recovery

- After a fast-retransmit set *cwnd* to *ssthresh/2*
  - i.e. don't reset *cwnd* to 1
  - Avoid unnecessary return to slow start
  - Prevents expensive timeouts
- But when RTO expires still do *cwnd* = 1
  - Return to slow start, same as Tahoe
  - Indicates packets aren't being delivered at all
  - i.e. congestion must be really bad

Fast Retransmit and Fast Recovery



- At steady state, *cwnd* oscillates around the optimal window size
- TCP always forces packet drops

Many TCP Variants…

BME-TMIT

- Tahoe: the original
  - Slow start with AIMD
  - Dynamic RTO based on RTT estimate
- Reno: fast retransmit and fast recovery
- NewReno: improved fast retransmit
  - Each duplicate ACK triggers a retransmission
  - Problem: >3 out-of-order packets causes pathological retransmissions
- Vegas: delay-based congestion avoidance
- And many, many, many more…

TCP in the Real World

BME-TMIT

- What are the most popular variants today?
  - Key problem: TCP performs poorly on high bandwidth-delay product networks (like the modern Internet)
  - Compound TCP (Windows)
    - Based on Reno
    - Uses two congestion windows: delay based and loss based
    - Thus, it uses a *compound* congestion controller
  - TCP CUBIC (Linux)
    - Enhancement of BIC (Binary Increase Congestion Control)
    - Window size controlled by cubic function
    - Parameterized by the time *T* since the last dropped packet

Things you should keep in mind about TCP…

- When you are a programmer
- When you are a network operator

- Why the difference?
  - As programmer you don’t care about the network between the endpoints
  - As operator, you don’t care about application needs

As a programmer…

1. There is traffic even before sending the first byte of data
   - The three-way handshake, blocking
2. Data arrives as in blocks
   - Unless push/urgent is used (rarely is used)
3. TCP throughput is affected by application buffer reads
   - Sometimes this is a good thing

TCP window

- Bandwidth delay product
- 8k – bandwidth is limited
  - Window is a limiting factor when delay is high
- 64K – maximum possible without options
  - better
  - Window scale option – scale up the window field

Bandwidth delay product

- Optimal value for window can be calculated
  - Win = RTT * Bandwidth
- Similarly,
  - Bandwidth = win / RTT
- Assuming that there is no other bottleneck in the network

TCP options example

No.	Time	Source	Destination	Protocol	Length	Info
4	0.168986	192.168.0.11	239.255.255.250	SSDP	175	M-SEARCH * HTTP/1.1
5	0.221892	fe80::d0f9:8c1:d62f:eb63	ff02::1:3	LLMNR	86	Standard query 0x7e01 A isatap
6	0.000117	192.168.0.11	224.0.0.252	LLMNR	66	Standard query 0x7e01 A isatap

```
▷ Frame 12: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
▷ Ethernet II, Src: Wistron_2d:ab:ba (00:1f:16:2d:ab:ba), Dst: 3Com_03:04:05 (00:01:02:03:04:05)
▷ Internet Protocol Version 4, Src: 192.168.0.11, Dst: 192.168.0.168
◢ Transmission Control Protocol, Src Port: 29385, Dst Port: 22, Seq: 0, Len: 0
    Source Port: 29385
    Destination Port: 22
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  ▷ Flags: 0x002 (SYN)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x822a [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ◢ Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
    ▷ TCP Option - Maximum segment size: 1460 bytes
    ▷ TCP Option - No-Operation (NOP)
    ▷ TCP Option - Window scale: 2 (multiply by 4)
    ▷ TCP Option - No-Operation (NOP)
    ▷ TCP Option - No-Operation (NOP)
    ▷ TCP Option - SACK permitted
  ◢ [Timestamps]
      [Time since first frame in this TCP stream: 0.000000000 seconds]
      [Time since previous frame in this TCP stream: 0.000000000 seconds]
```

As a programmer… (cont)

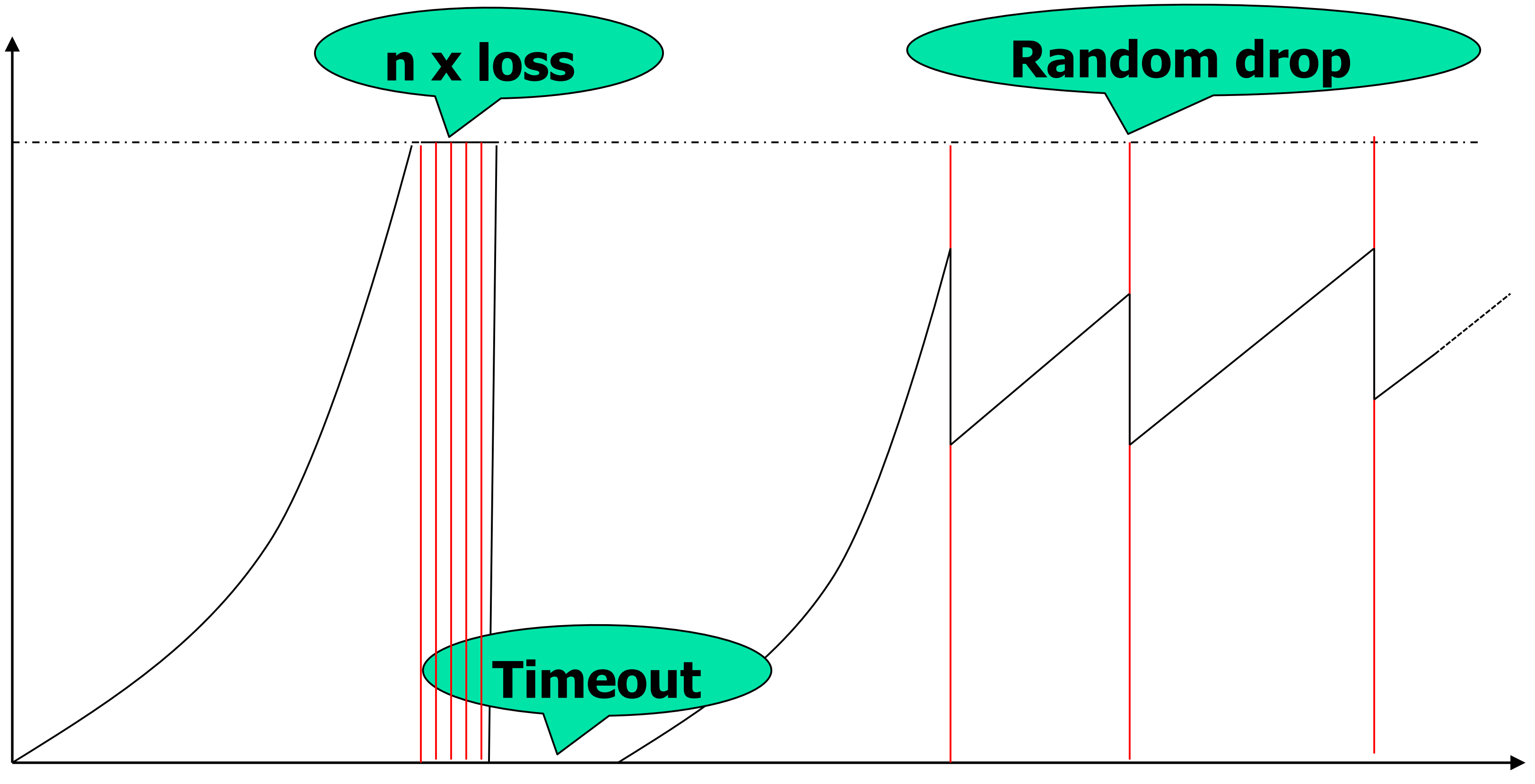
4. Bandwidth may be limited by the network
   - Problem when a required bandwidth is not avail
5. Socket options
   - Some TCP algorithm behavior can be changed
6. Reuse of addresses
   - Fin bit – close of connection – still lingering

As network operator…

1. Type of traffic generated
   - Bursty by nature
2. Bottleneck detection
   - Traffic never goes over ~80% (aggregate)
3. Congestion control algorithms in routers
   - To handle congestion in advance
   - Achieve fairness
4. Lossy channels- radio
   - Misinterpreted as bottleneck

RED – Random Early Drop

- Buffer management in routers
  - A single loss is better than a timeout



TCP - Wireless

- Problem:
  - Wireless – loss due the radio
    - no bottleneck!
    - TCP misunderstands, reduces the cwnd
  - Solutions
    - WTCP – proxy
    - SACK – selective acknowledgements

Thank You!

- End -

MŰEGYETEM 1782
Budapest University of Technology and Economics
Department of
Telecommunications and Media Informatics