CS3.301 Operating Systems and Networks

TCP Explained and Introduction to Memory Virtualization

Karthik Vaidhyanathan

https://karthikvaidhyanathan.com

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

SERC
Software Engineering Research Centre

TCP is the most used protocol on the internet. How does TCP work?

What all you need to provide some features that TCP provides?

A Small Analogy

Network and Link Layer

Communication Channel cannot be always reliable!!

Communication Channel

What can we do from the protocol perspective?

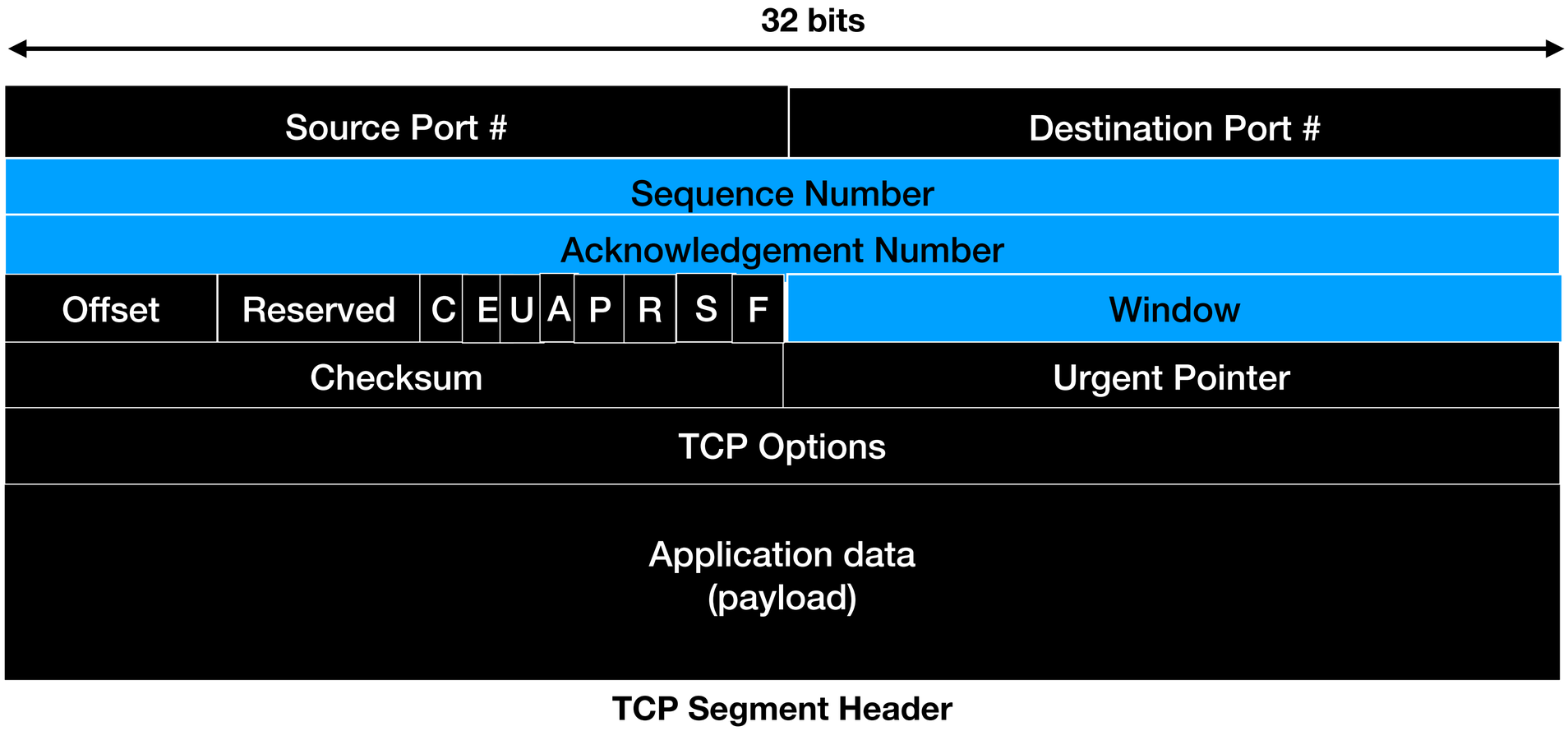
Person 1 Talking
(Process in a host A)

Both speak the same language and have similar speed in talking

Person 2 Talking
(Process in a host B)

Do we foresee some challenges?

Lets go into TCP - Header



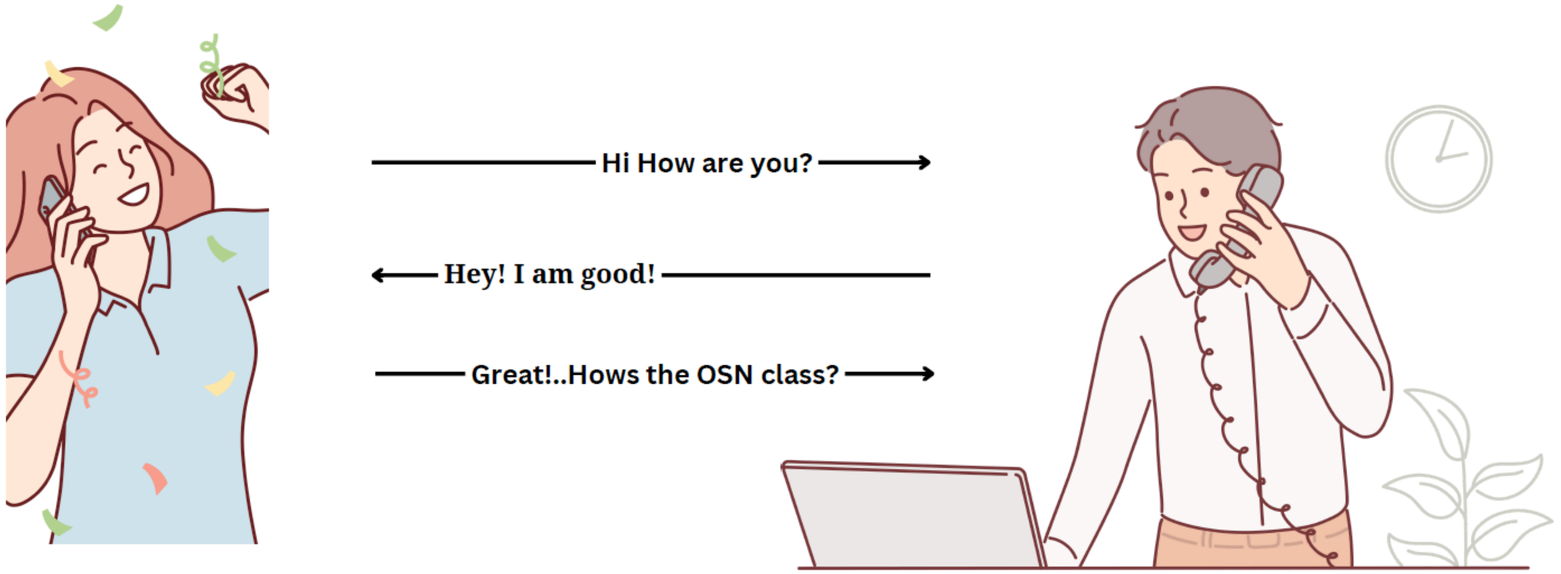
TCP Segment Header

Header Elements

- **Sequence number:** Tracks bytes that are sent (# of bytes that are sent)
- **Acknowledgement number:** Tracks bytes that are received (Sequence number of the next expected byte)
- **Window/Receive Window:** Number of bytes the receiver can accept (Flow control)
- **A:** Acknowledgement bit
- **R, S, F:** Connection management
- **C, E:** Congestion notification
- **Offset:** Length of the TCP header

What do ACK and Sequence Number do?

Reliability!!



Person 1 Talking
(Process in a host A)

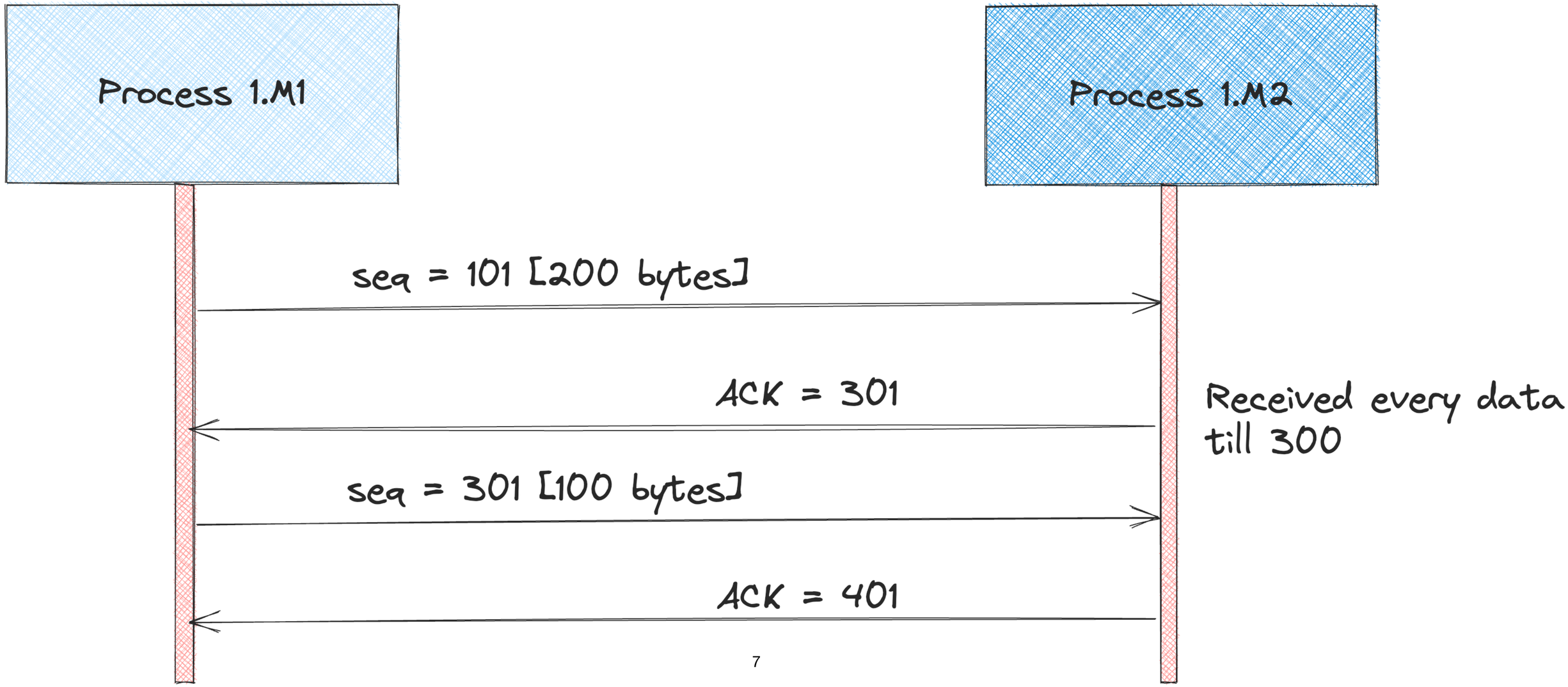
Person 2 Talking
(Process in a host B)

Each word the Person 1 says reaches person 2 in the same order

Whatever Person 1 Says, Person 2 acknowledges before adding new points to the conversation

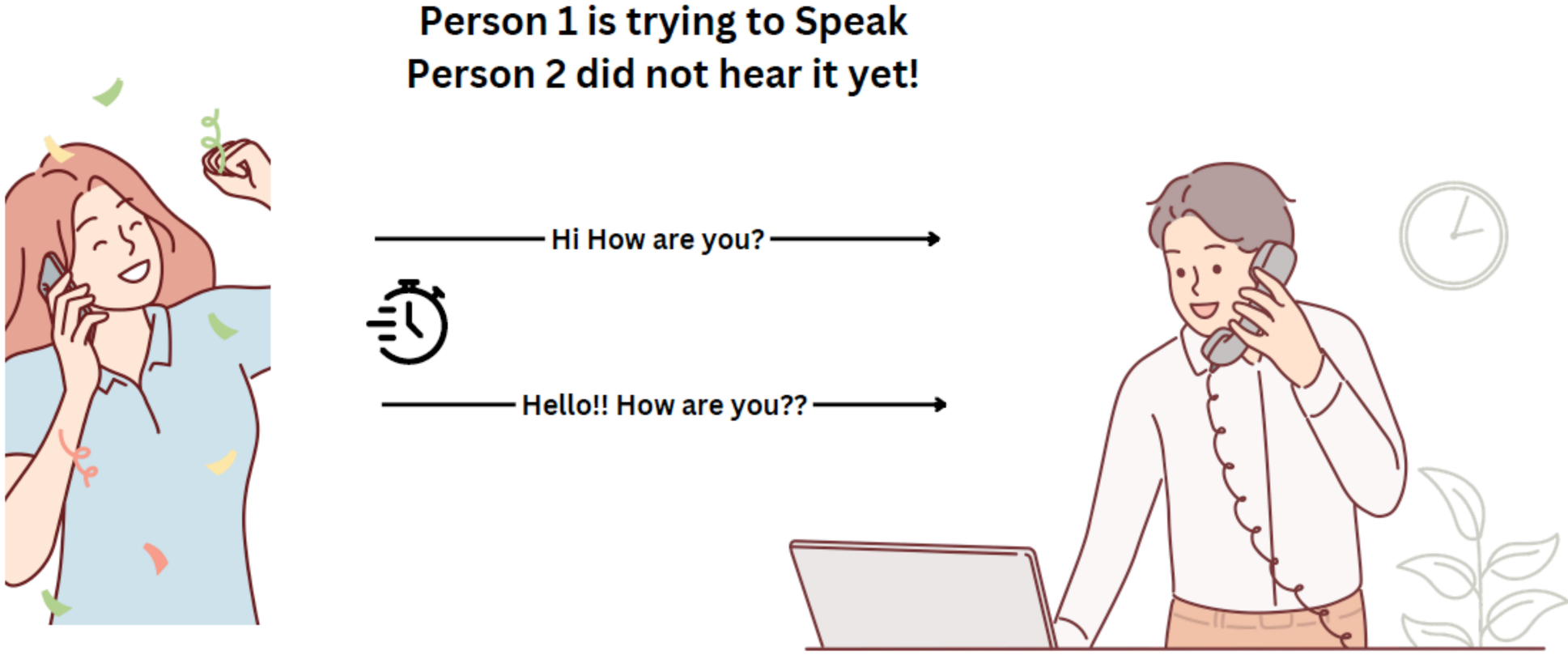
What do ACK and Sequence Number do?

Reliability!!



How to handle if data is lost?

Can we retransmit?

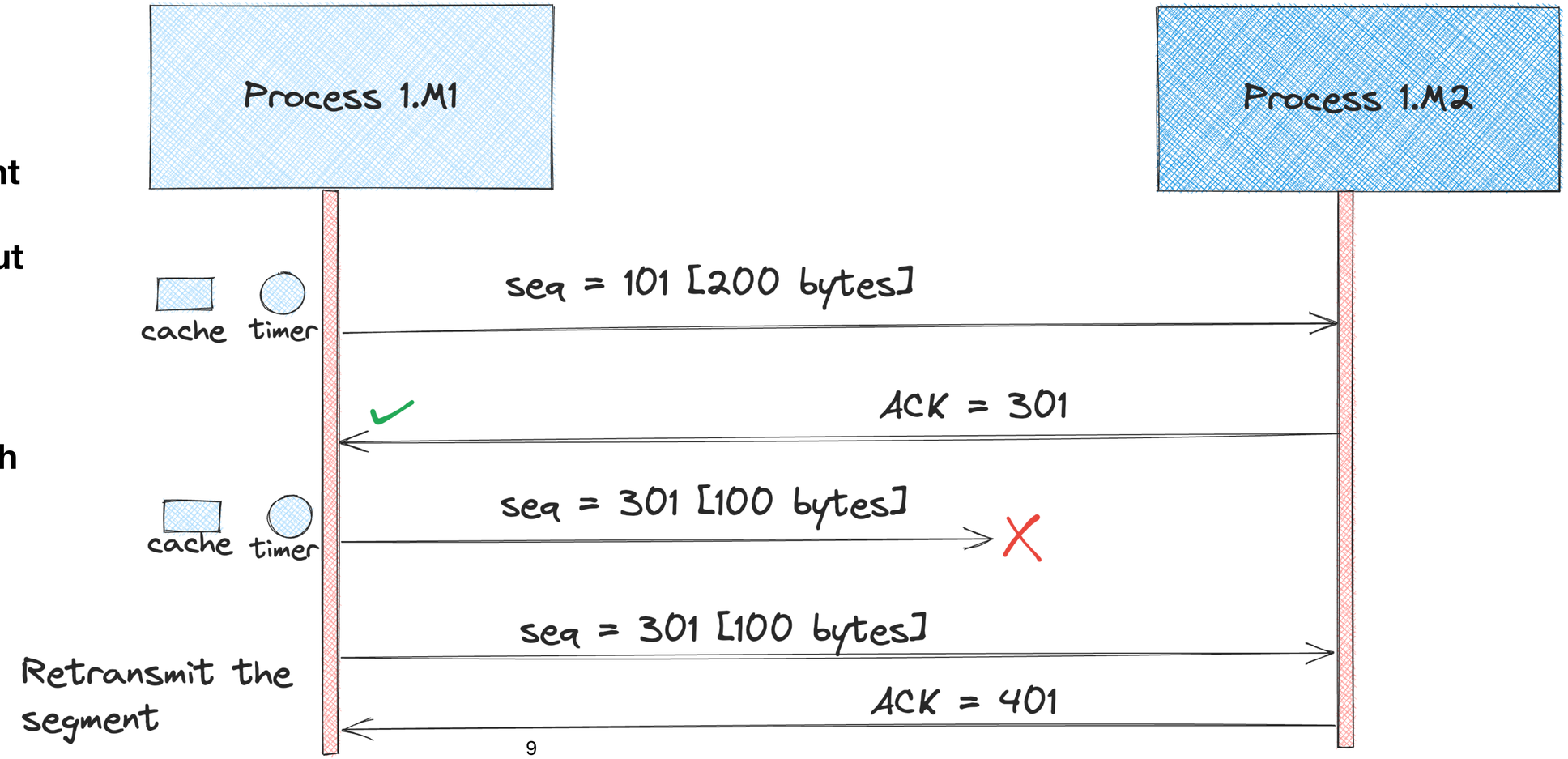


How to handle if data is lost?

Retransmission timeout also known as Round Trip Timeout (RTT)

TCP caches every data sent in a buffer (OS supports) Until retransmission timeout

What if ACK does not reach Back Process 1.M1?



How to calculate RTT?

$$\text{EstimatedRTT} = (1-\alpha)*\text{EstimatedRTT} + \alpha*\text{SampleRTT}$$

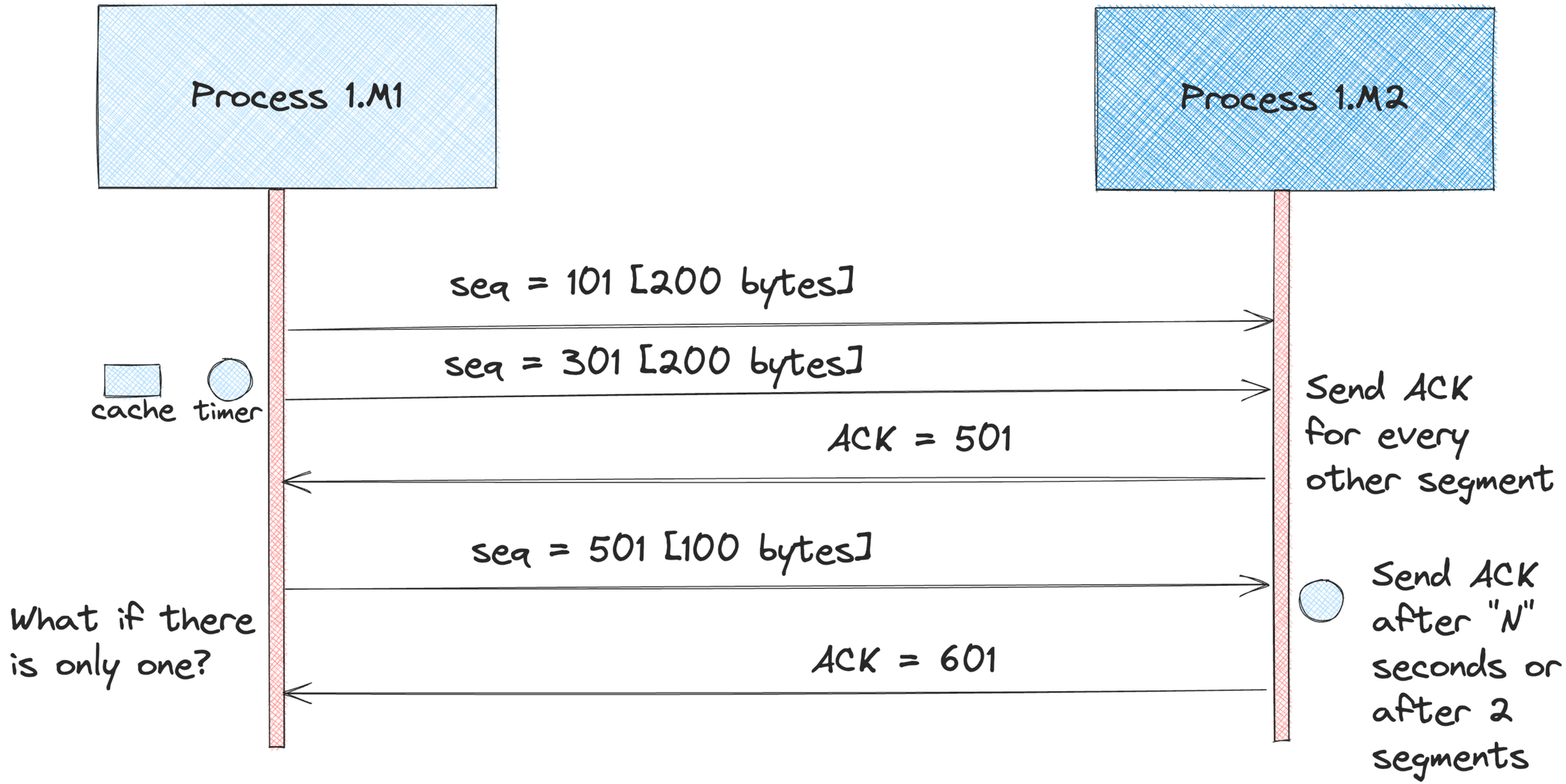
$$\text{DevRTT} = (1-\beta)*\text{DevRTT} + \beta*|\text{SampleRTT}-\text{EstimatedRTT}|$$

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4*\text{DevRTT}$$

- **SampleRTT:** Time measured from segment transmission until ACK receipt
- **EstimatedRTT:** Estimated weighted moving average (EWMA) $\alpha = 0.25$
- **DevRTT:** EWMA of sampleRTT deviation from EstimatedRTT $\beta = 0.75$
- **TimeoutInterval:** Estimated Time plus some kind of safety margin

Do We need to Send ACK for each segment?

Use delayed acknowledgements



What if the speed is high?

Person 2 is speaking fast and many things..Person 1 is not getting time to process

Hi How are you? →

← I am good how are you?

← I was wondering that.....

← Also there is one more thing...

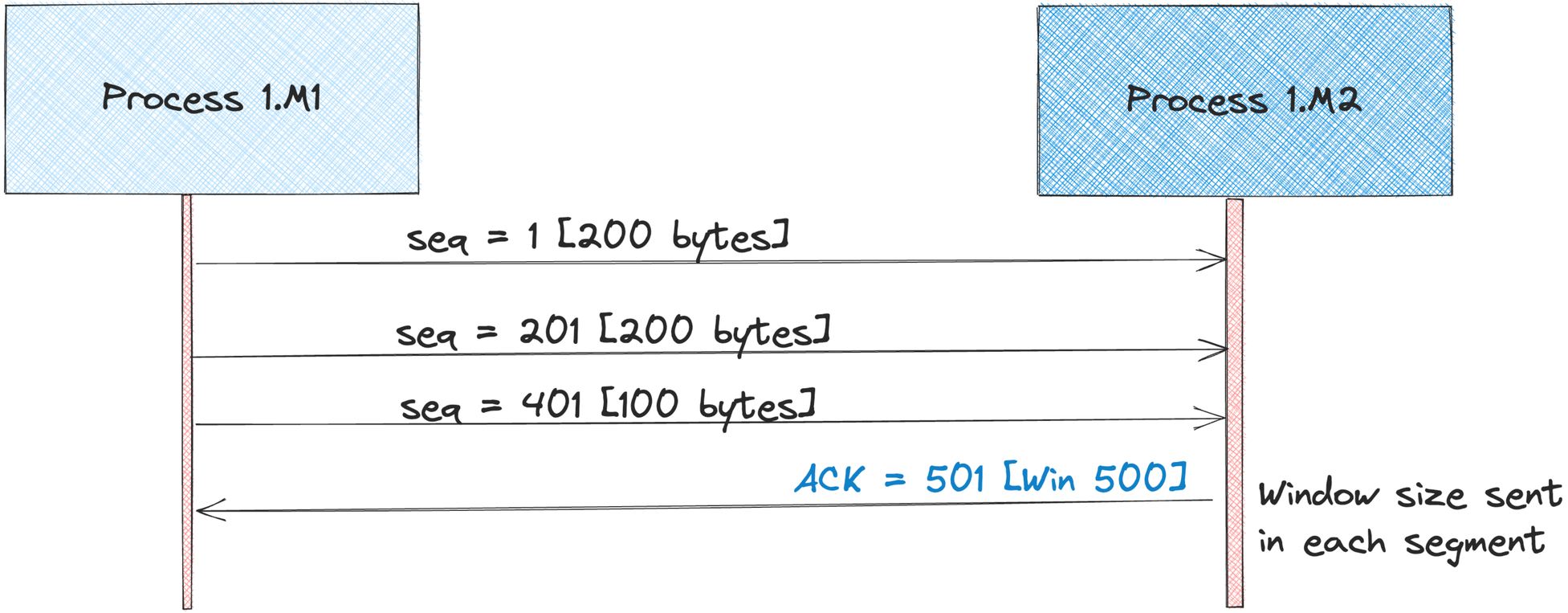
Can you please speak slowly? →

Person 1 Talking
(Process in a host A)

Person 2 Talking
(Process in a host B)

Sending too much data is also problem

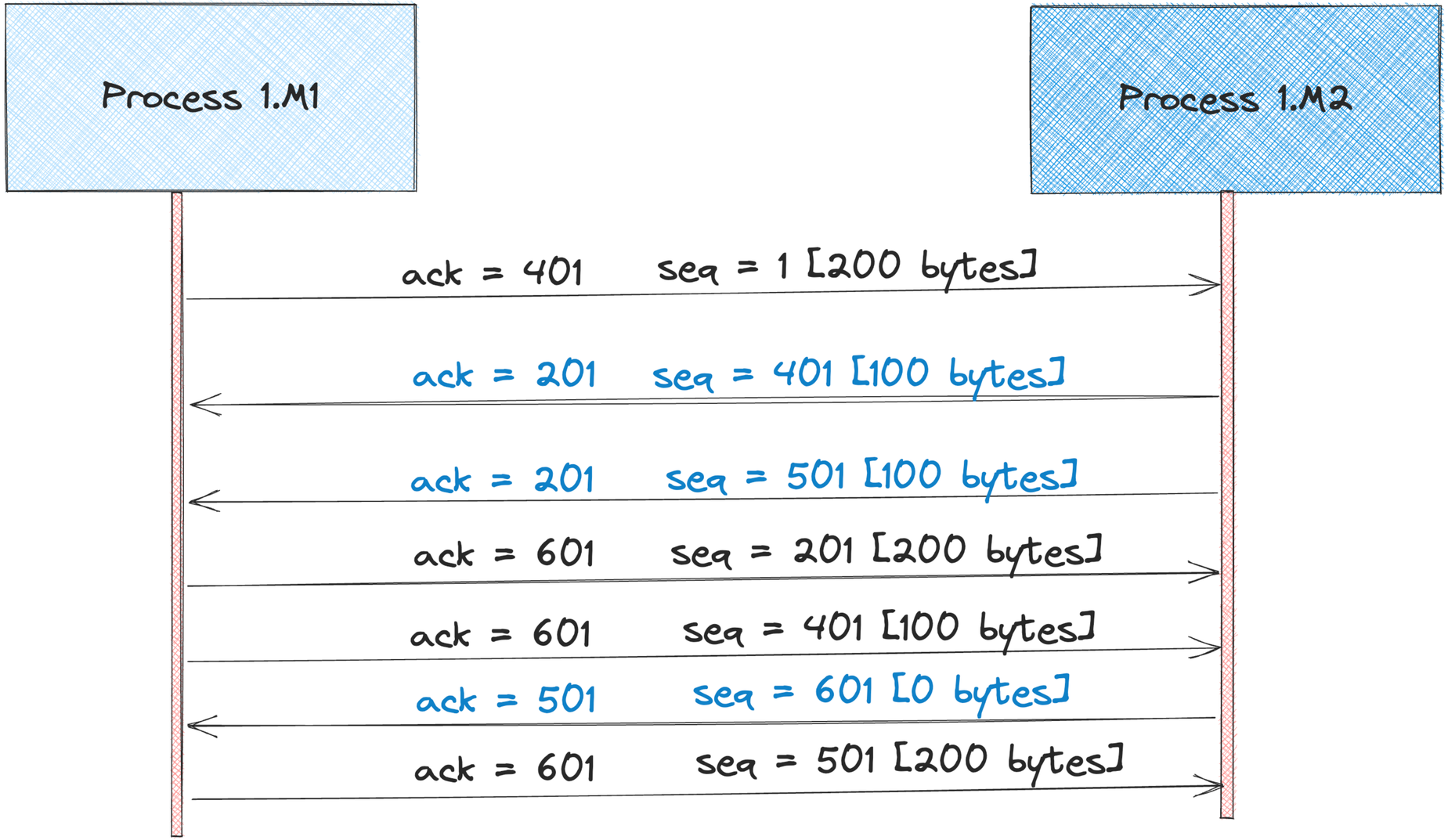
Window Size - Flow Control



- Dynamic update of Window size will enable flow control
- What if Process 1.M2 sends a windows size of 0?

TCP is bidirectional

Both Senders can send data

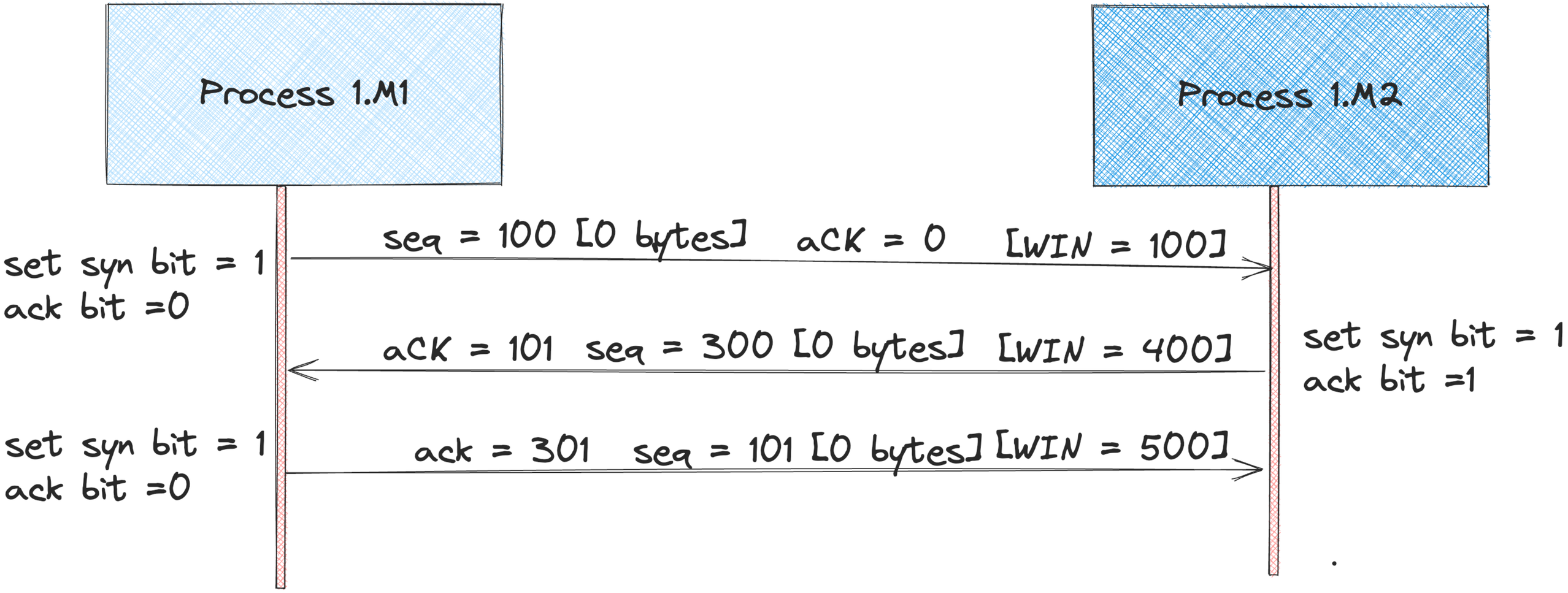


How to choose sequential numbers?

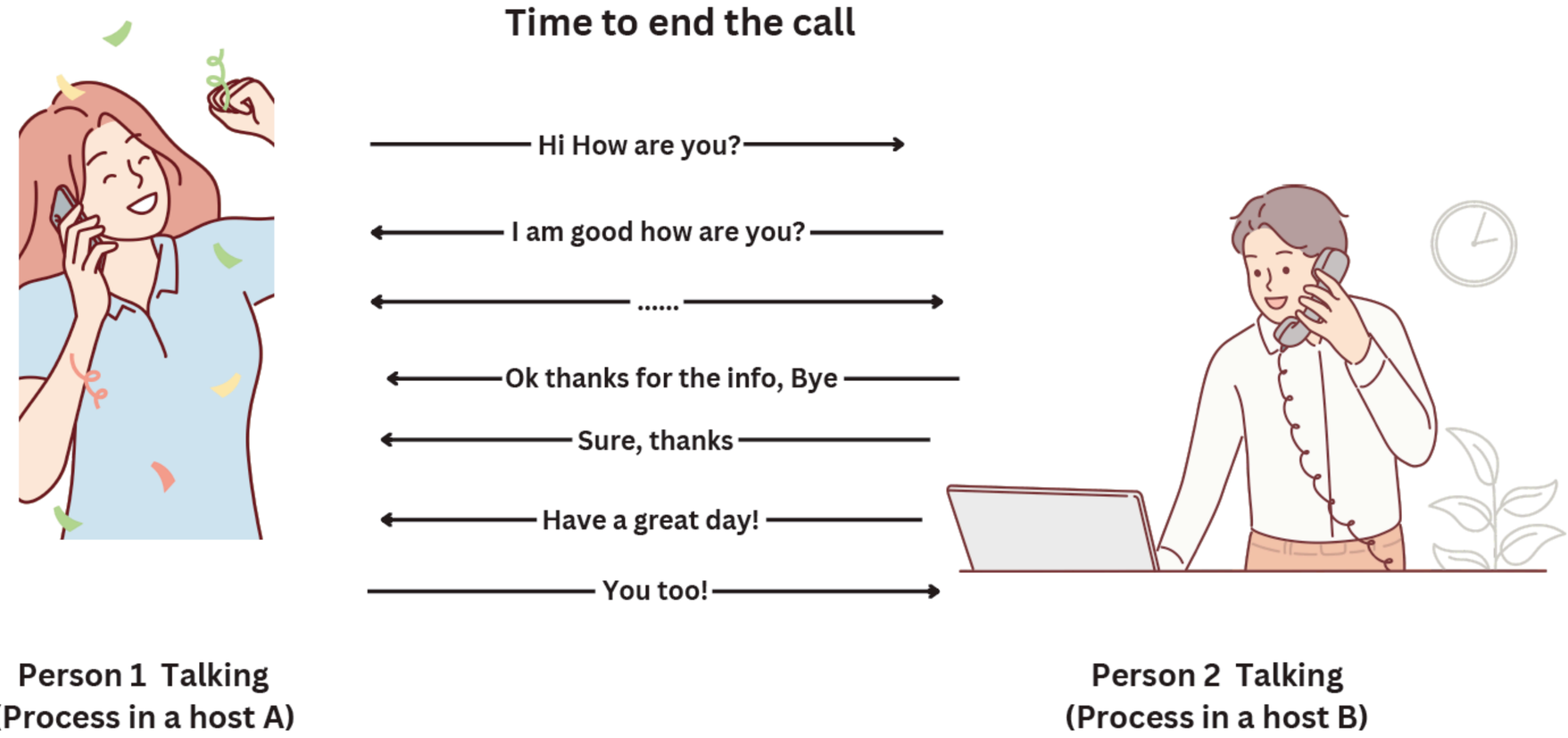
- Initial sequence numbers are randomly chosen by the senders
- Each can select a sequence number during the connection establishment
- Connection establishment in TCP happens through 3-way handshake
- The 3-way handshake consist of 4 events:
  - Process 1.M1 sends a connection request with SYN bit set and sequence number of X
  - Process 1.M2 acknowledges the connection request and sends back an ACK with X+1
  - Process 1.M2 also sends a request with the SYN bit set and sequence number [Y]
  - Process 1. M1 acknowledges the receipt by sending ACK [Y+1]

Three Way Handshake

Establishing Connection



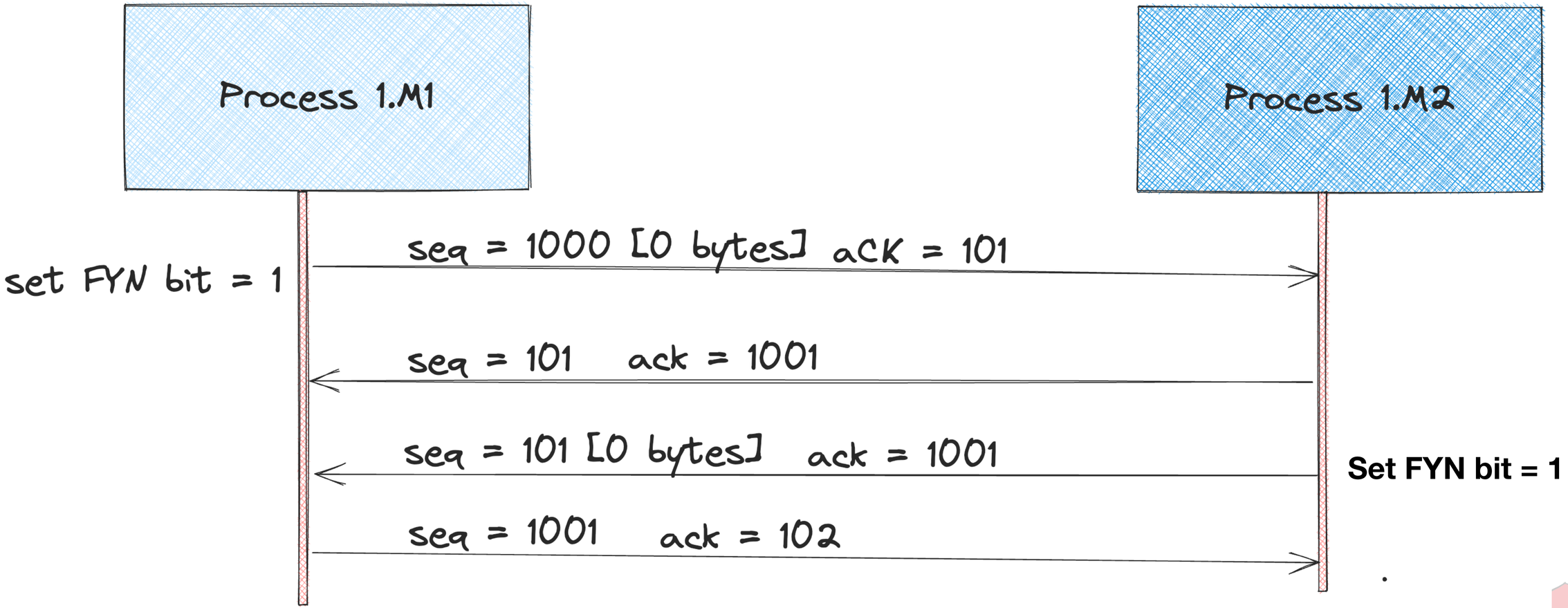
Closing Connection



- TCP has two ways to close connection: **FIN** and **RST** flags

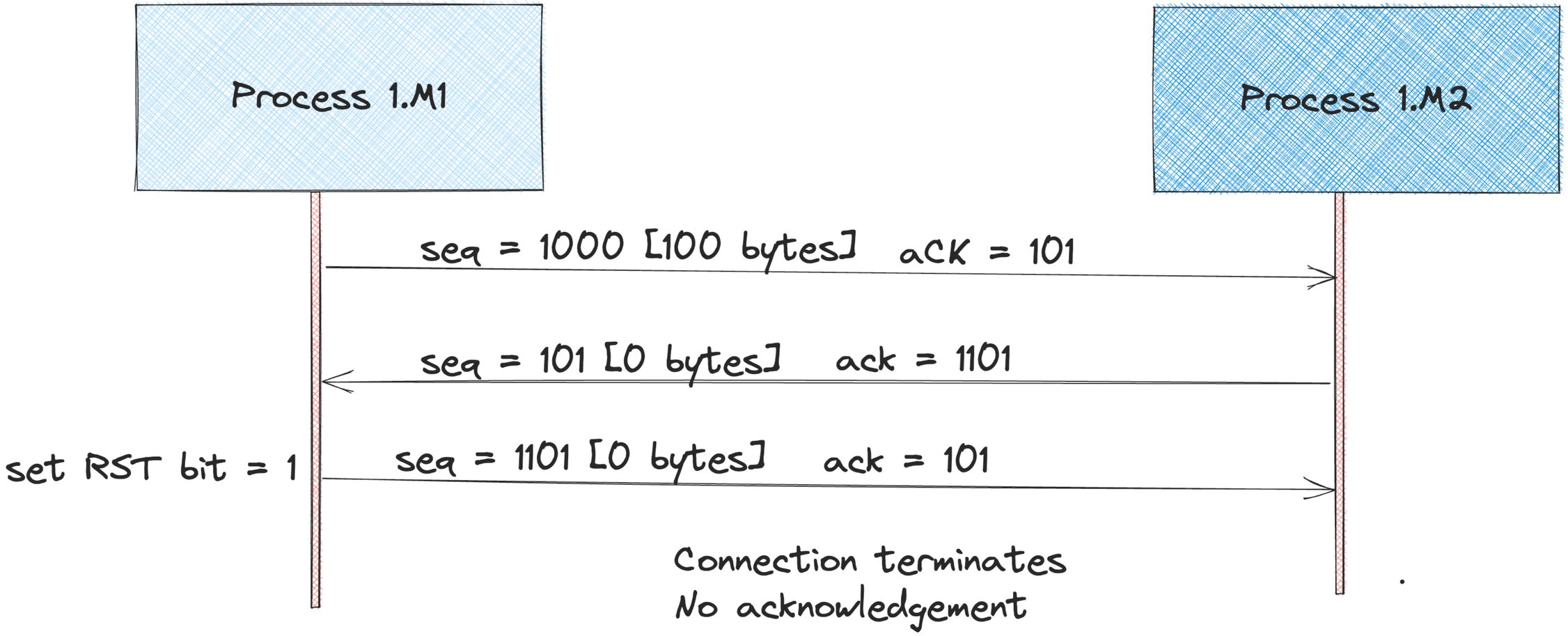
Using FYN bit

Graceful termination (Four-way closure)



Using RST Flags

Ungraceful closing



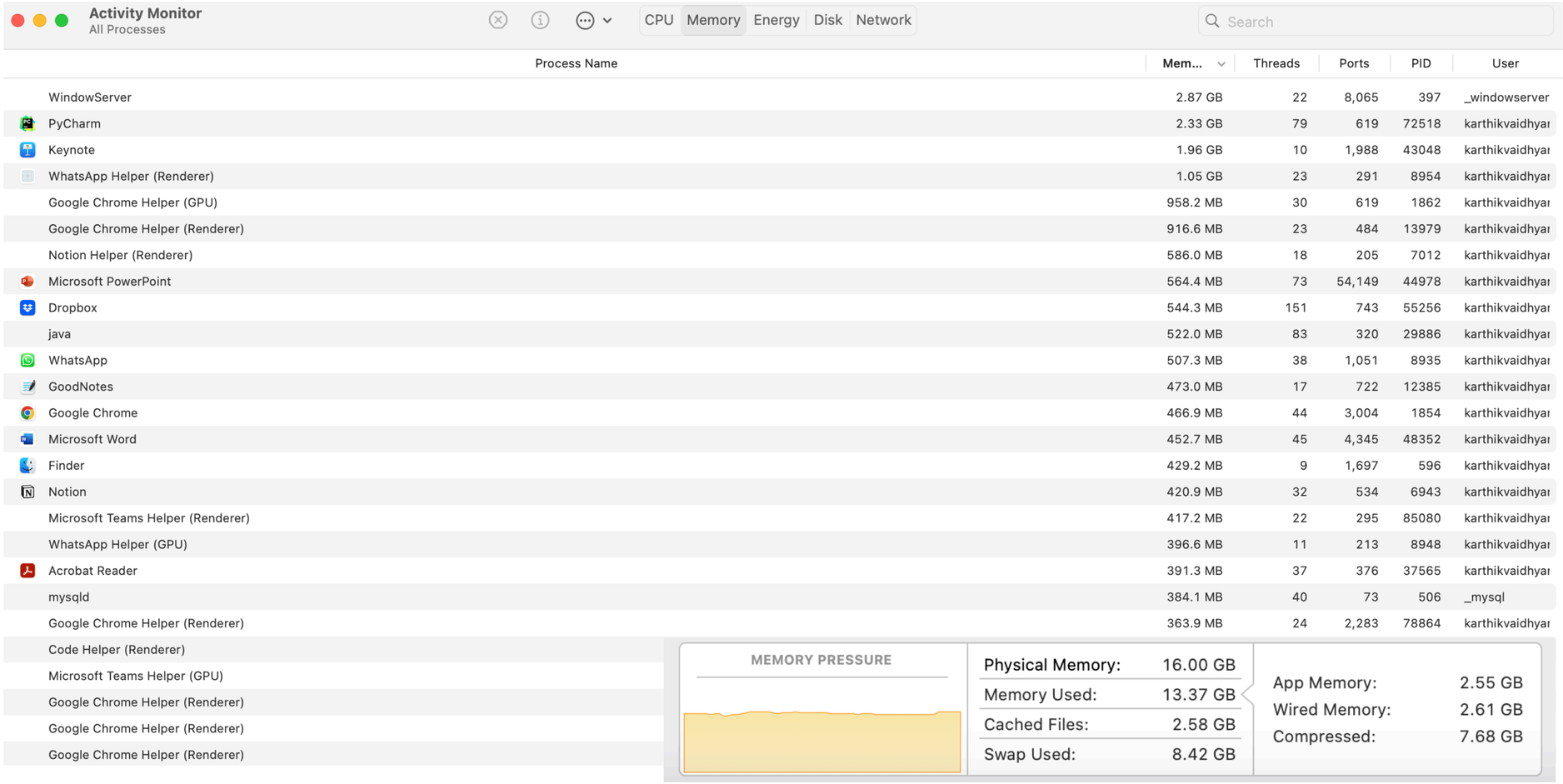
But we need Memory!

How does OS handle the memory requirements of all these?

Where is the process stored? What about network buffer?

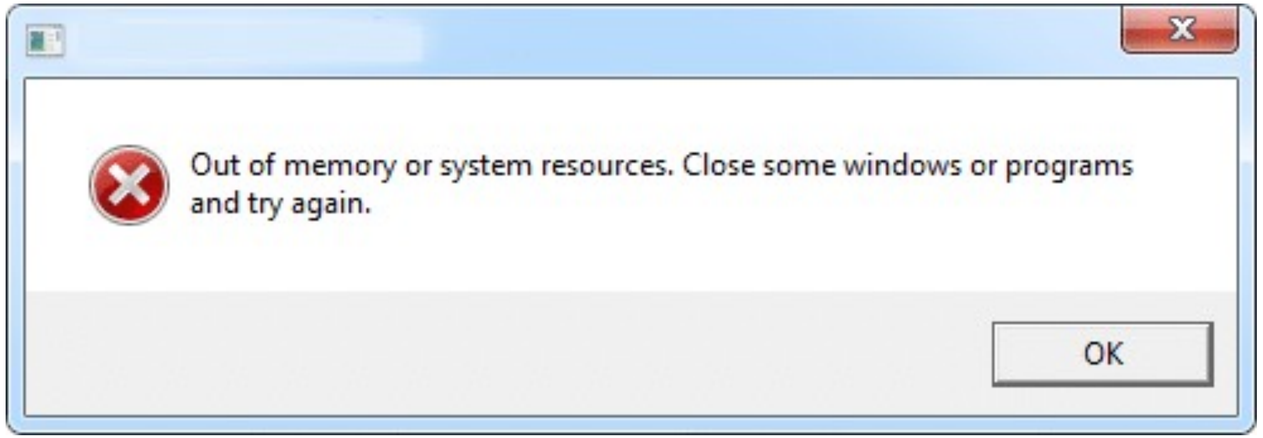
Many processes run at the same time!

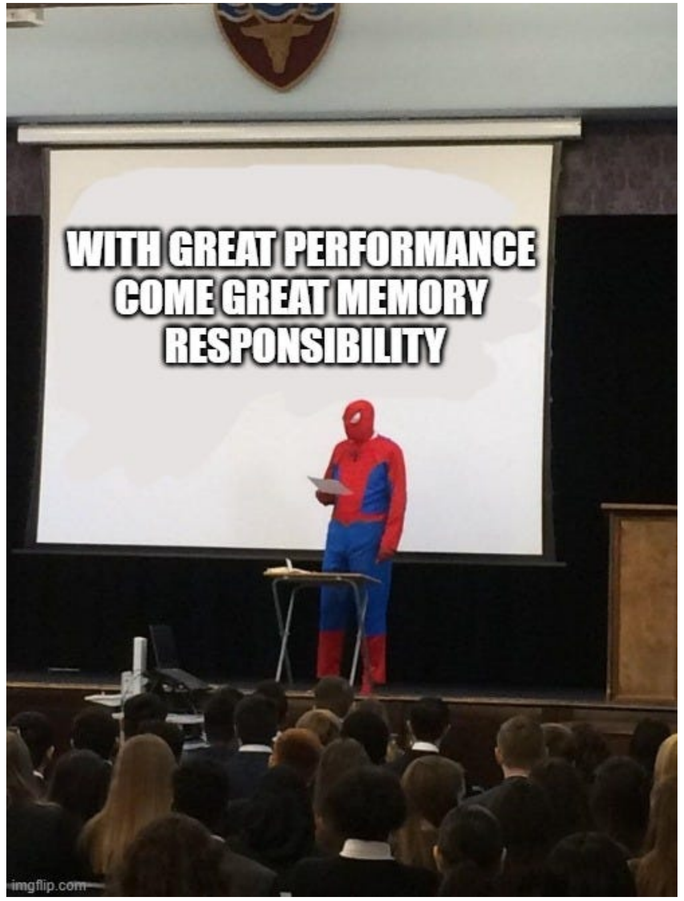
- What about Memory? Do we have enough Memory?



Real View of Memory can be Messy!

Managing it can be even further difficult

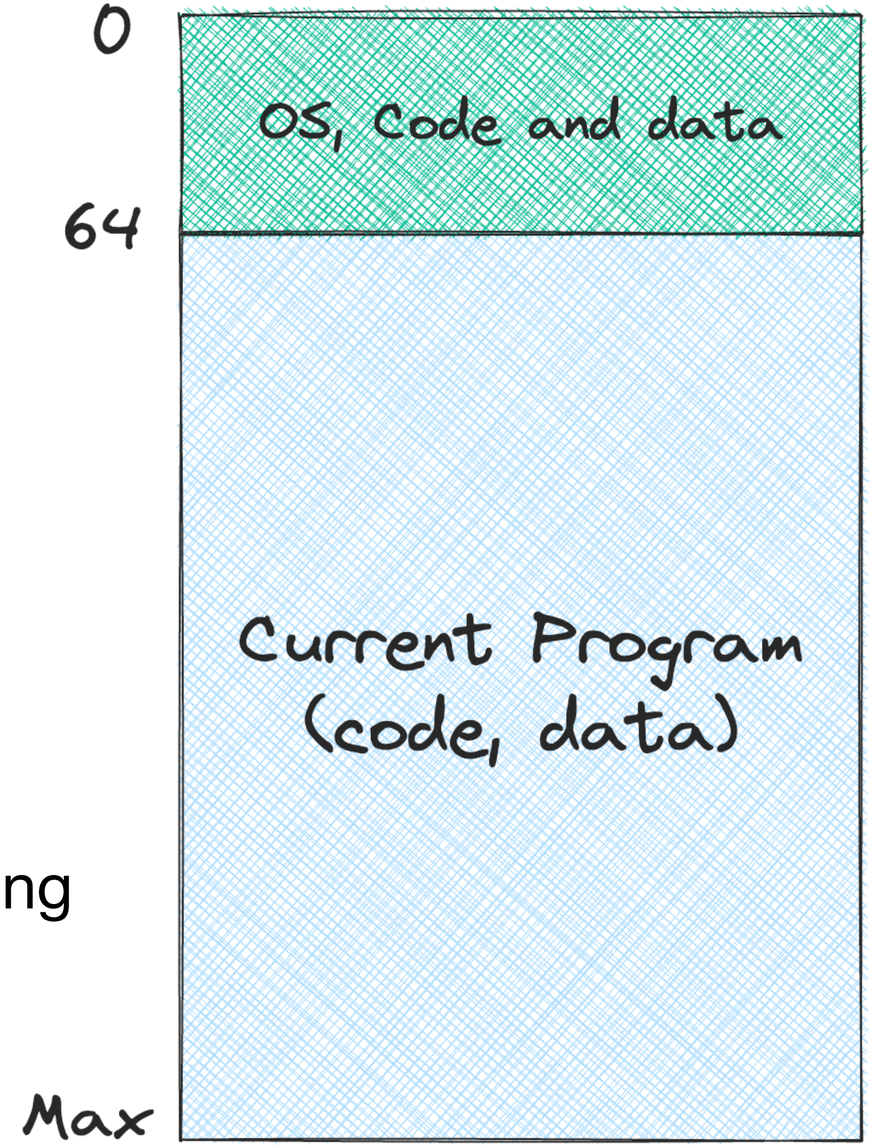




Source: Google Images

Memory Virtualization

- Early days OS had just one program
- OS, its code and data resides in one part
- The running program, its code and data resides in one part
- Does it work today?
  - Today its about multiple processes
  - Run process for sometime save everything to disk, run next - **Problems?**
- OS provides process virtualisation



Only One running Program

Memory Virtualization: Why?

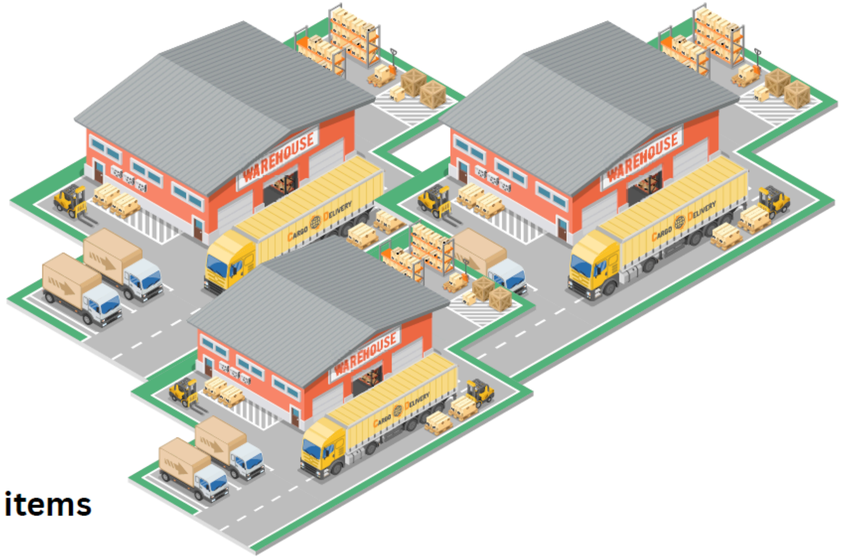
- We need to think about multiple processes
- Need to increase utilisation and efficiency
- Particularly useful in olden times when it costed millions of dollars for machines
- Soon came era of time sharing
- Batch computing was not anymore appreciated
- Instead of saving in the disk, can we keep the process on disk itself?

An Analogy

Onsite Shopping

Every users have access to different items but to a limited set

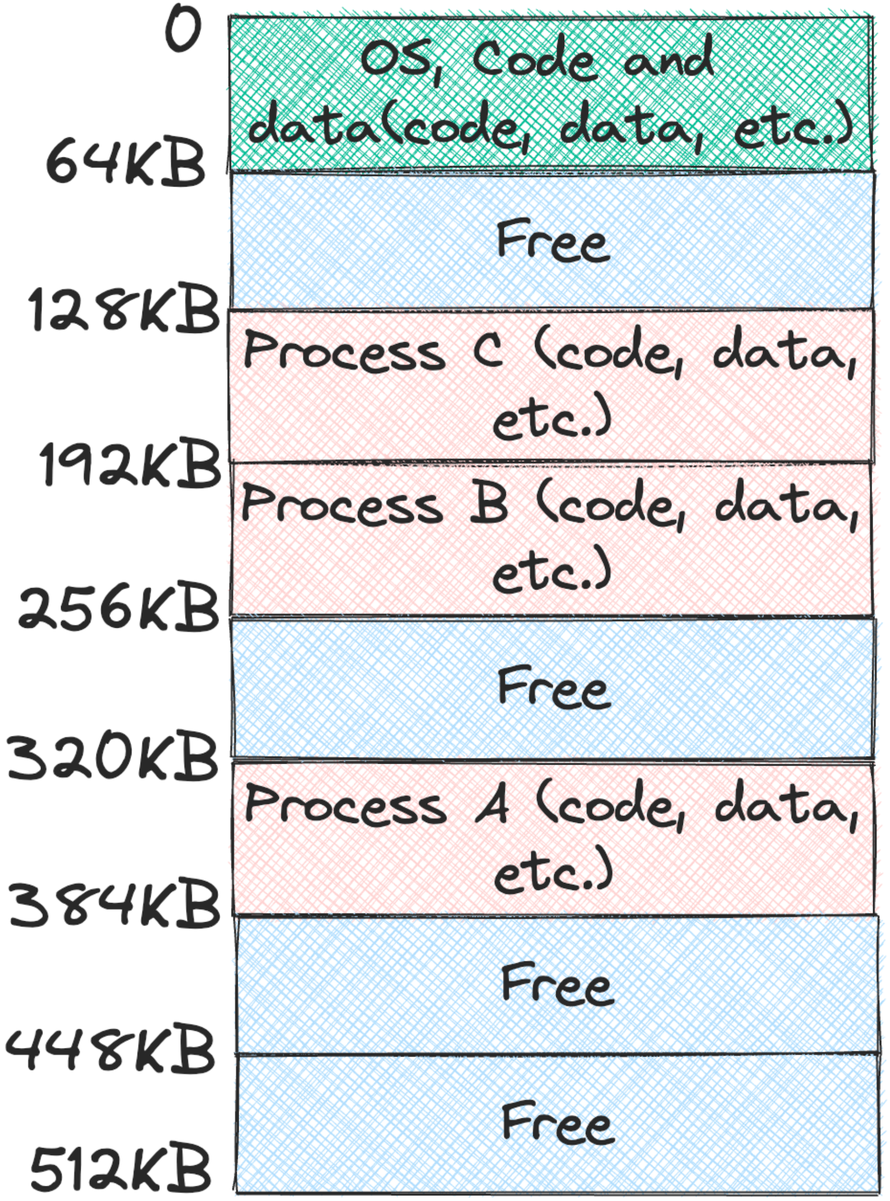
Online Shopping



Every Users feel that they have access to infinite items

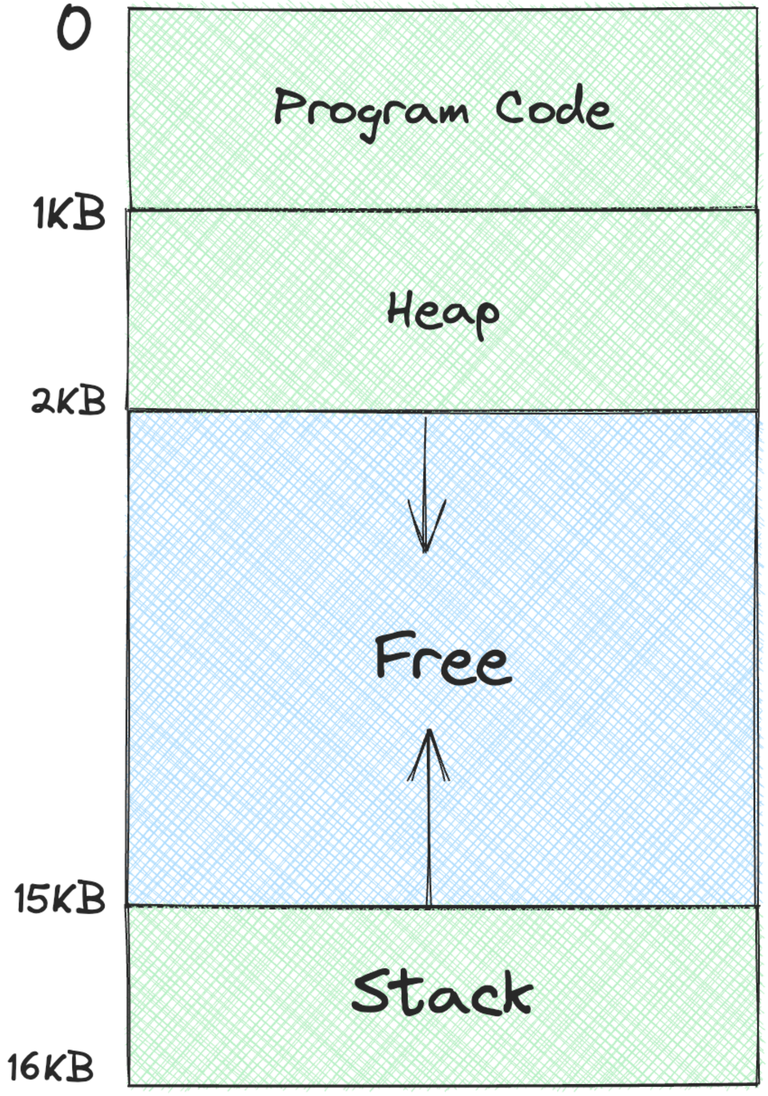
Keep Process in the Memory

- Each process is given a dedicated location
- There are multiple free spaces where process can be added
  - Main challenge: We don’t want any process to read any other process data
  - Real life OS has 100s of process that will be running
  - Giving control to user may make it hard



Abstraction: Virtual Address Space

- OS creates easy to use abstraction of the physical space
- Address space (Memory image of process)
  - Program Code (and static data)
  - Heap - Dynamic memory allocations (malloc)
  - Stack - Function calls during runtime
  - The stack and heap grow during runtime
- Every process assumes that it has access to large block of memory from 0 to MAX
- CPU issues loads and stores to virtual addresses

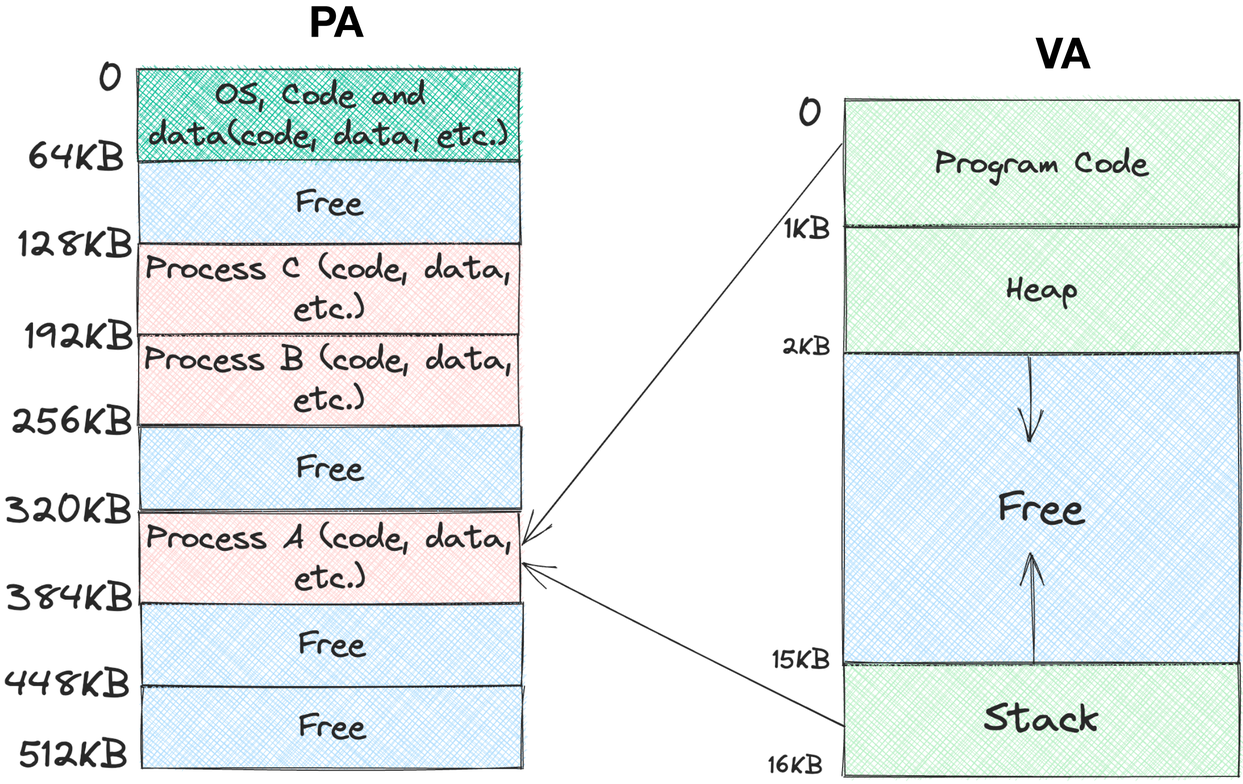


There is only one physical memory

- How can OS build the abstraction of a private large address space on top of single physical memory?
  - There is only one physical memory, process feels has it has its own starting at 0
  - When a process tries to load from a particular location, **K** (0)
    - OS with some hardware support ensures that the load doesn't go to actual location
    - Rather to the physical address **Z** (320) - **Virtualization**

How actual memory is reached?

- Address translation from virtual address (VA) to physical address (PA)
  - CPU loads/stores to VA but memory hardware access PA
- OS allocates memory and tracks the location of the process
- Translation is done by Memory Management Unit (MMU)
  - OS makes necessary information available



Thank you

Course site: karthikv1392.github.io/cs3301_osn
Email: karthik.vaidhyanathan@iiit.ac.in
Twitter: @karthi_ishere