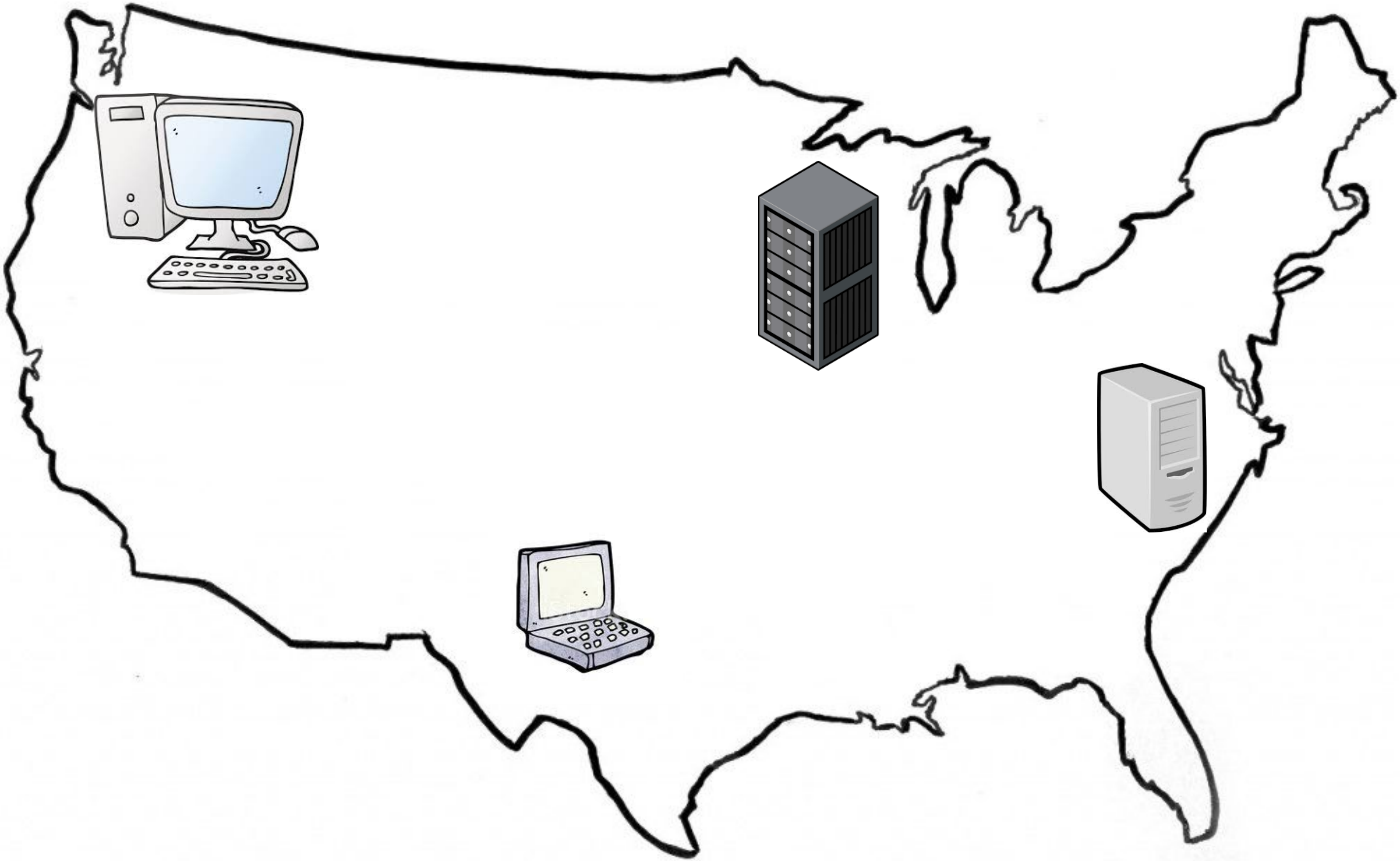
UPPSALA UNIVERSITET

Reliable communication and the protocol stack

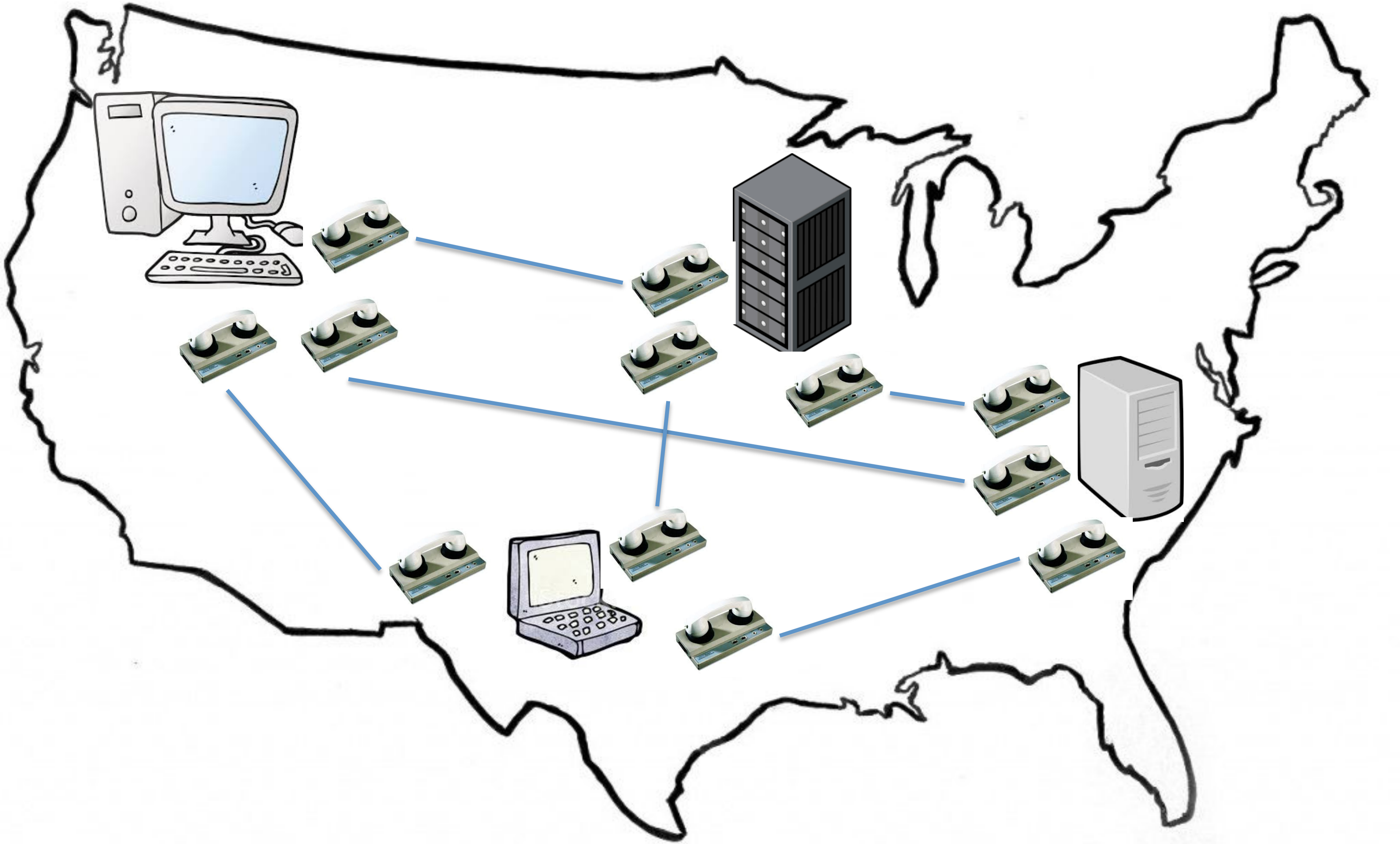
Lars-Åke Nordén

Origins of the Internet

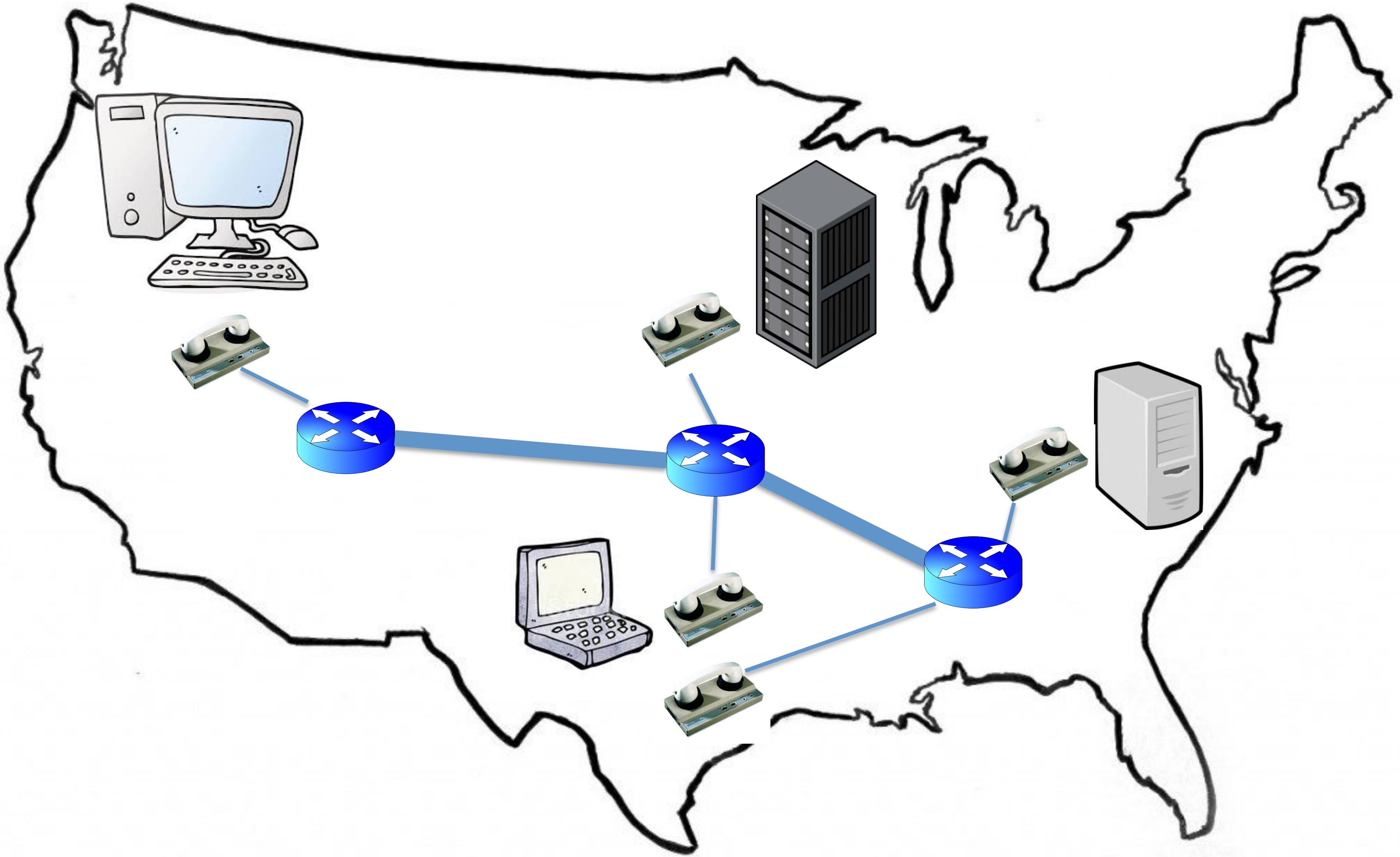


UPPSALA UNIVERSITET

Pre-Internet communication

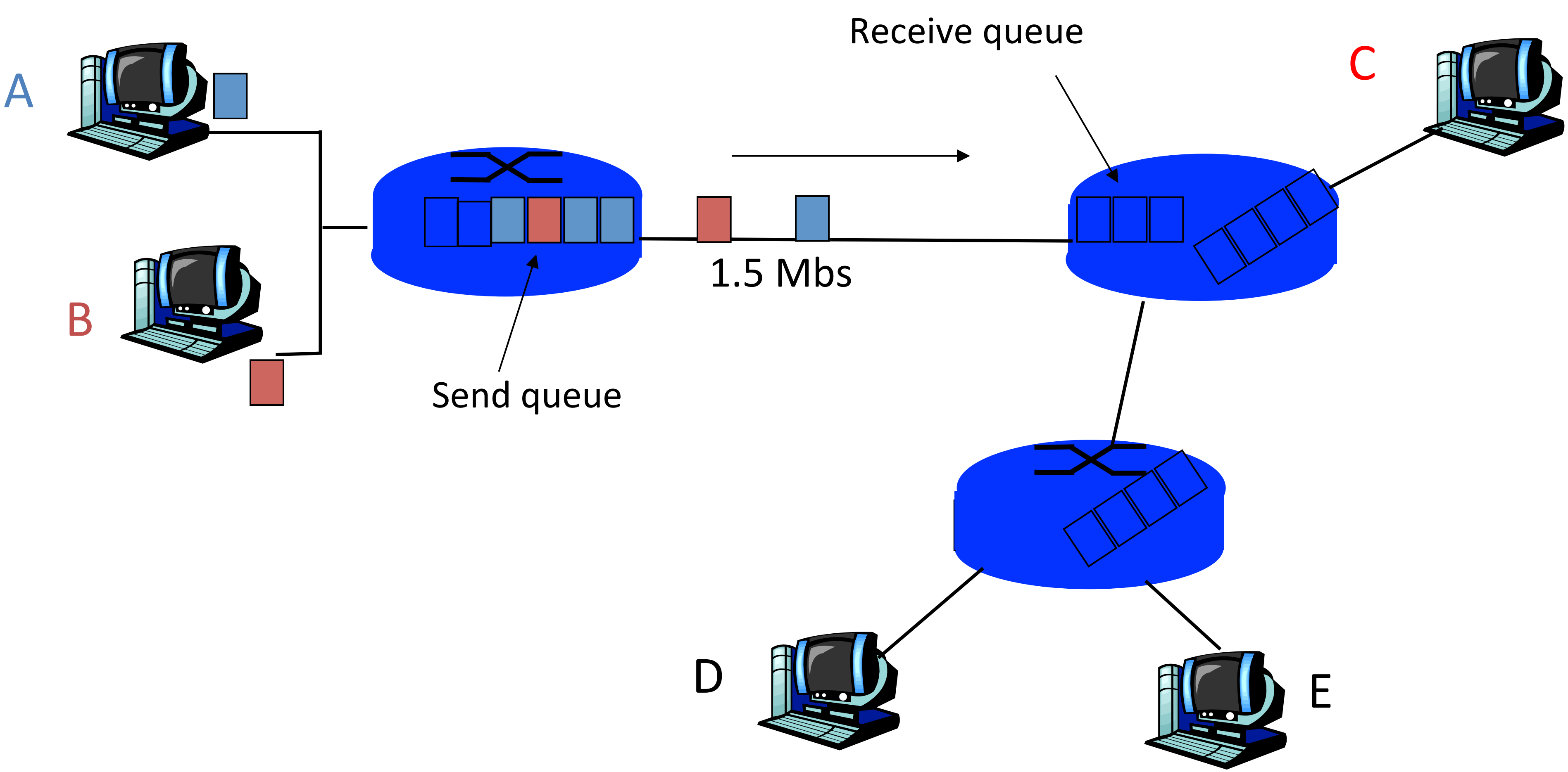


Idea of statistical multiplexing



UPPSALA UNIVERSITET

Statistical multplexing



Properties of statistical multiplexing

- More efficient usage of communication links
  - Resource sharing
- Queues, queues, queues
  - To handle short-time peaks in forwarding needs

Two main types of networks that use statistical multiplexing

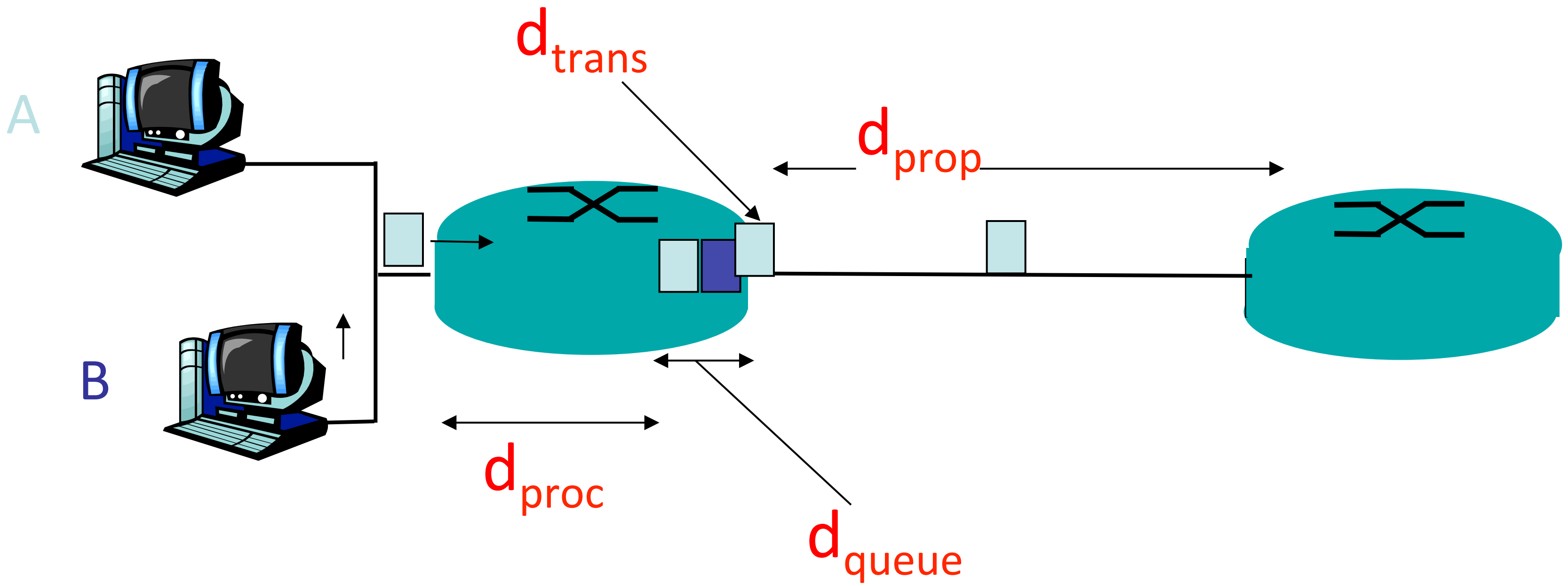
Virtual circuits

- The path for a message is determined before the message is sent
- Routers are keeping track of the path, forwarding the message to the next node

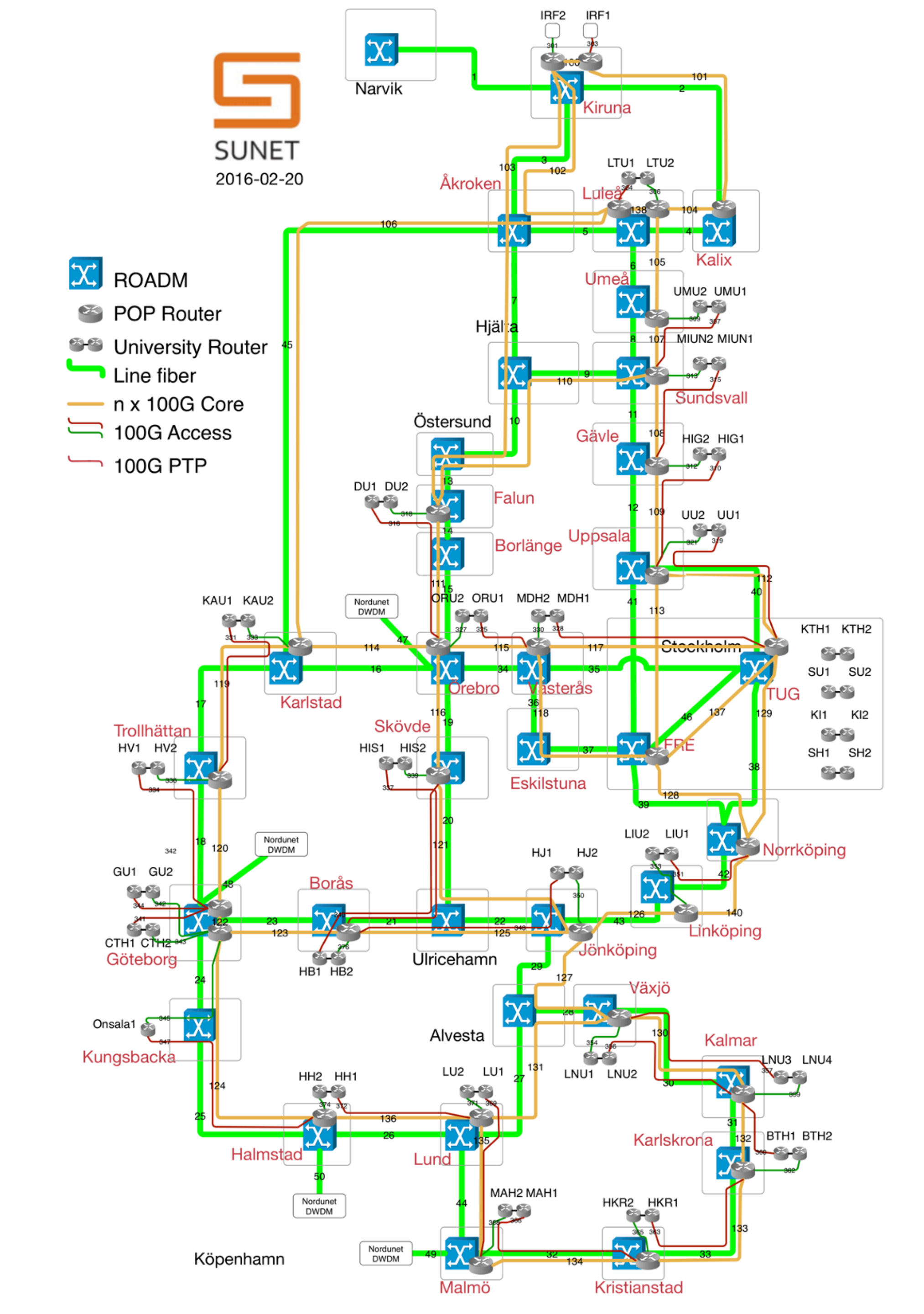
Datagram networks

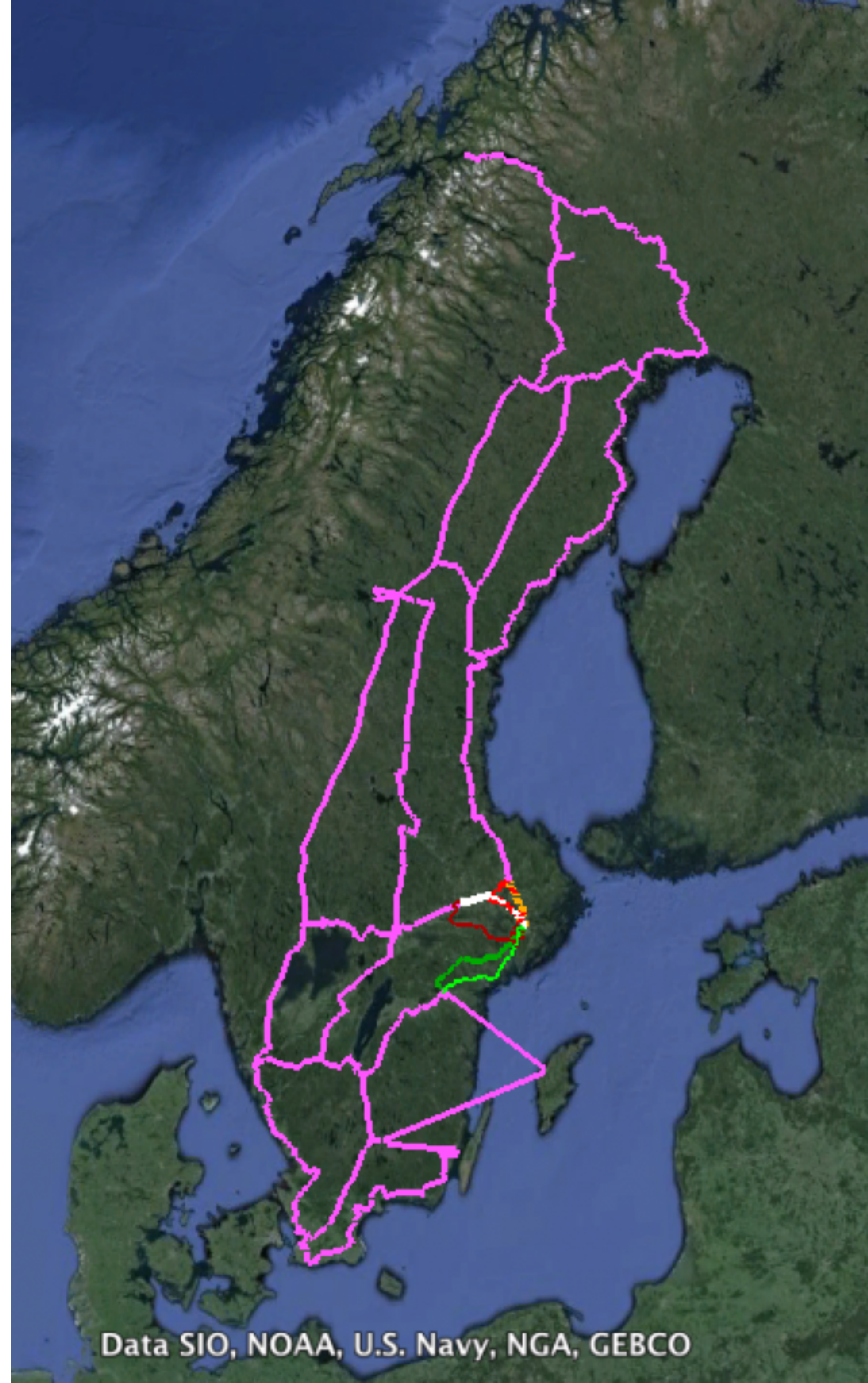
- Each message contains a destination address
- The next hop is decided on a per-message basis with the destination address as input
- The path from sender to receiver *may* vary with each message sent

Delay components



What affects the different delay components?


SUNET
2016-02-20
ROADM
POP Router
University Router
Line fiber
n x 100G Core
100G Access
100G PTP
Narvik
IRF2
IRF1
Kiruna
Åkroken
LTU1
LTU2
Luleå
Kalix
Umeå
UMU2
UMU1
Hjälta
MIUN2
MIUN1
Sundsvall
Östersund
Gävle
HIG2
HIG1
DU1
DU2
Falun
Borlänge
Uppsala
UU2
UU1
KAU1
KAU2
Nordunet DWDM
ORU2
ORU1
MDH2
MDH1
Stockholm
KTH1
KTH2
SU1
SU2
KI1
KI2
SH1
SH2
Karlstad
Örebro
Västerås
TUG
Trollhättan
HV1
HV2
Skövde
HIS1
HIS2
Eskilstuna
FRE
Norrköping
LIU2
LIU1
HJ1
HJ2
GU1
GU2
Borås
Linköping
CTH1
CTH2
Göteborg
HB1
HB2
Ulricehamn
Jönköping
Växjö
Onsala1
Alvesta
Kalmar
Kungsbacka
LNU1
LNU2
LNU3
LNU4
HH2
HH1
LU2
LU1
Halmstad
Lund
Karlskrona
BTH1
BTH2
MAH2
MAH1
HKR2
HKR1
Köpenhamn
Malmö
Kristianstad


Data SIO, NOAA, U.S. Navy, NGA, GEBCO

Best-effort datagram delivery

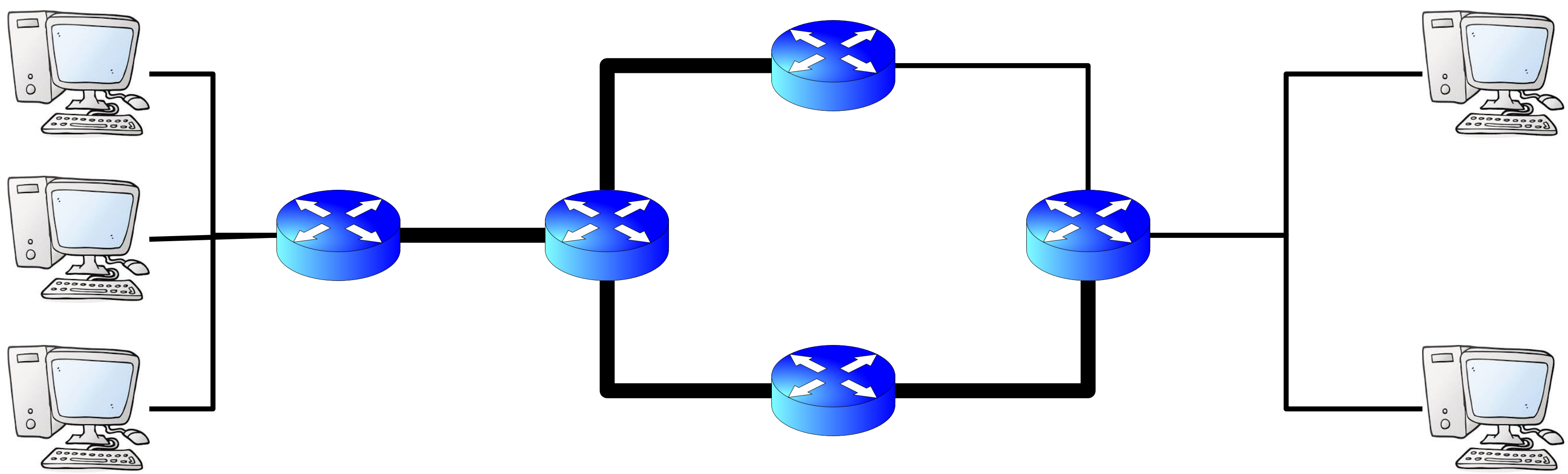
- Fundamental communication service of the Internet
  - Provided by the Internet Protocol (IP)

- Unit of communication: *packets*

- Packets may:
  - Disappear without any notice
  - Be exposed to unpredictable variations in delay
  - Be “damaged” upon reaching the receiver
  - Arrive out-of-order in comparison to sending order
  - …

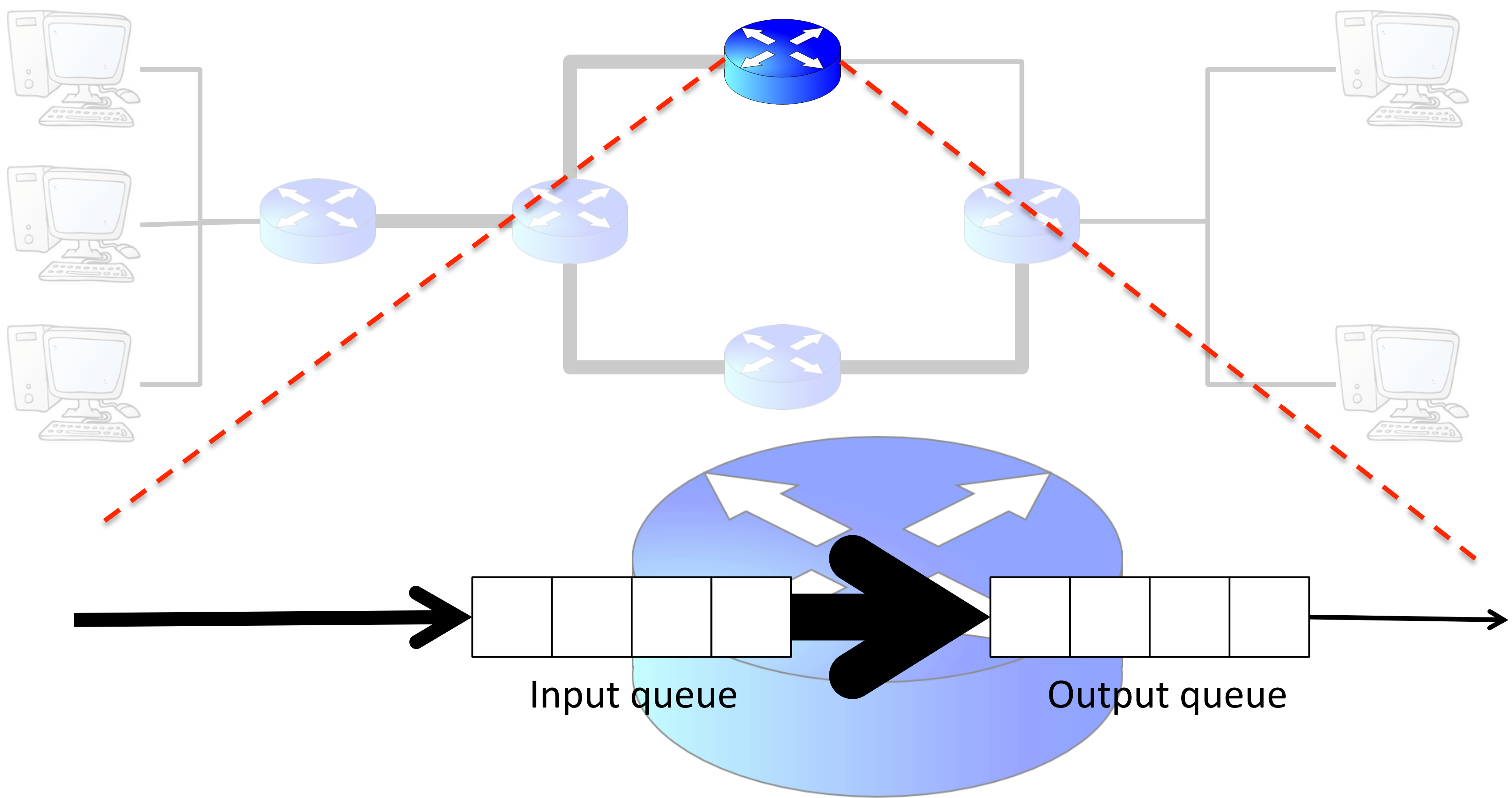
Why are there losses?

Because of statistical multiplexing and a best-effort service!



Why are there losses?

Because of statistical multiplexing and a best-effort service!



Queues in the Internet

- What happens if…
  - A queue is too small?
  - A queue is too large?
  - An input queue fills up faster than the output queue is emptied?

- What properties must data traffic have to avoid queueing problems?

Reliable communication

A sends data to B

- How does A know that B has received the information?
- How does B ensure that received information is processed in the right order?
- Should we use proactive or reactive reliability mechanisms?

Data loss in a network

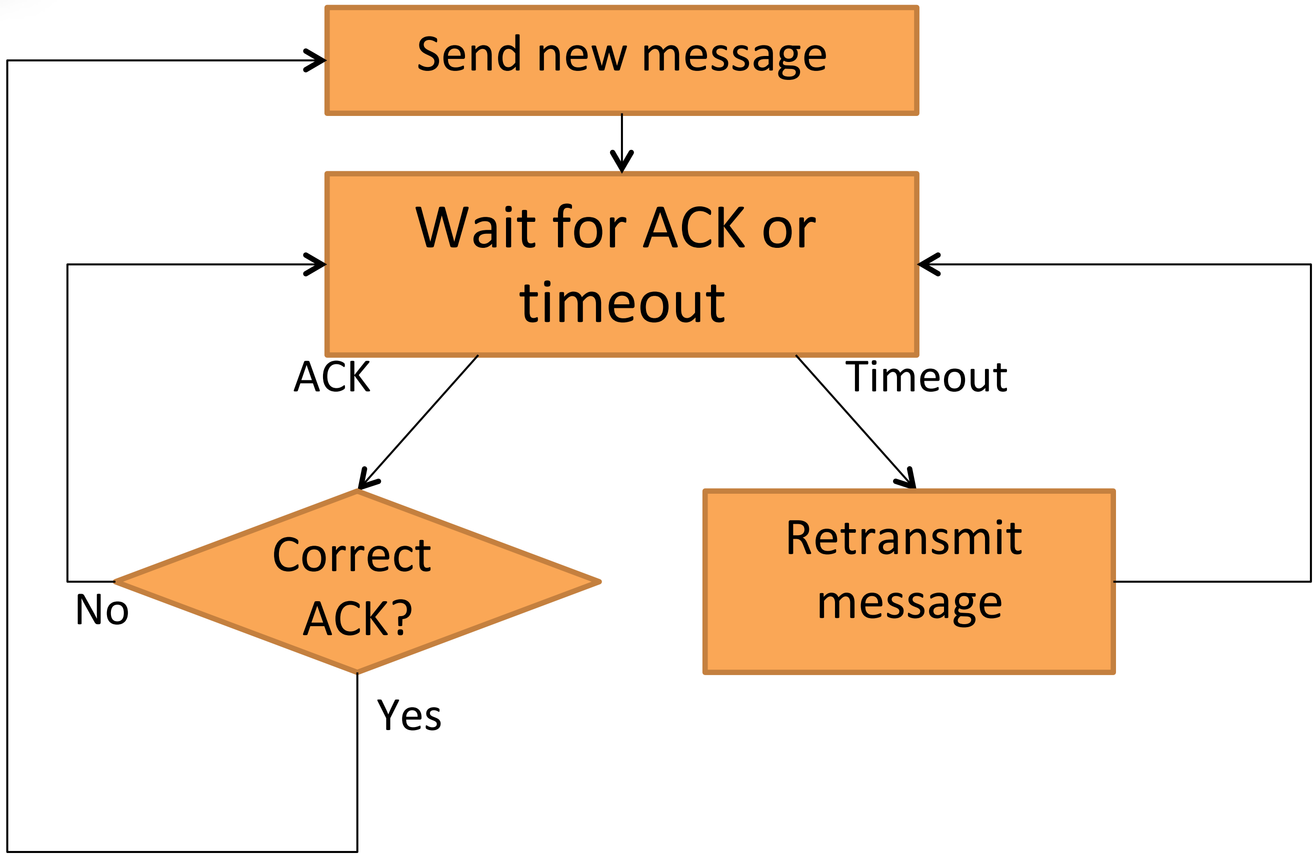
- Why is data lost in a network ?
- How can data loss be detected ?
- How can data loss be handled
  - Reactively ?
  - Proactively ?
- Different types of ARQ schemes

Introducing: Acknowledgments

- Sends information back to sender

- Can be of different types
  - Selective
  - Cumulative
  - Vector
  - Negative

ARQ scheme: Stop-and-wait



Stop-and-Wait

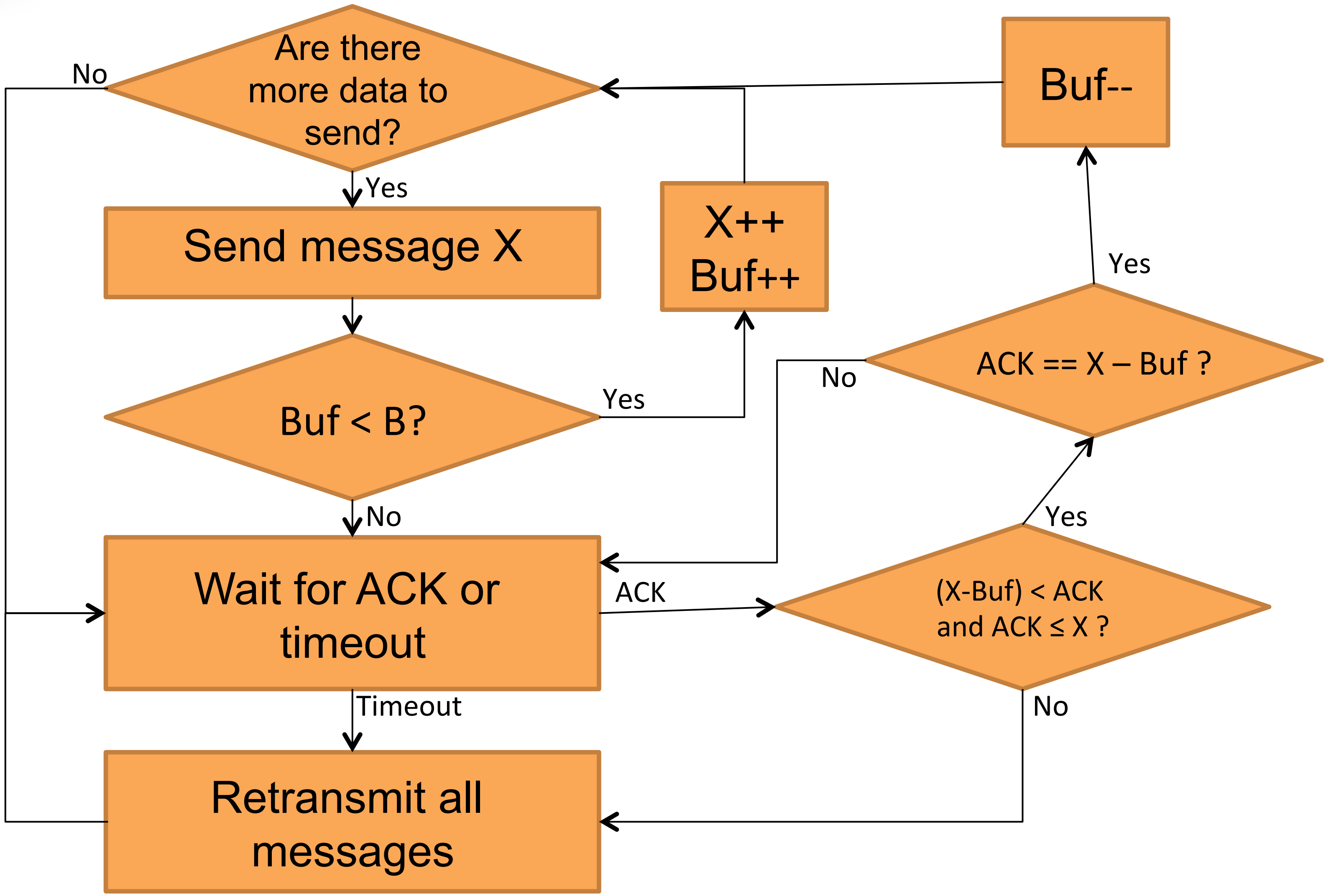
- Slow
  - Much waiting time

- Underutilizes the connection
  - Connection idle at least 50% of the time

- Can we pipeline?

ARQ: Go-Back-N with buffer size B

Variable *Buf* set to 1 at start



Go-Back-N

- Retransmits entire buffer
  - Can cause congestion
  - Not very clever

ARQ: Selective-repeat

- Mark ACK:ed messages as received even if it was not the expected ACK
- Only retransmits the message that caused a timeout
- “Forgets” sent messages when there are consecutive ACKs from the beginning of the sender’s “window”

Reordering

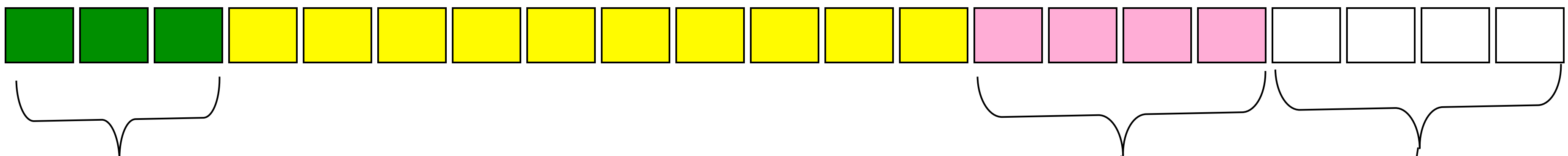
- Maintain a “window” at the receiving end
- Put incoming messages in the right slot in the window
- Deliver messages to client when there are consecutive messages at the beginning of the “window”

Sliding window - Sender side

Cumulative Acknowledgments

Not sent | Sent, no ACK | ACK:ed | Free

Sending buffer at the sender:



New data sent to transport layer by application, but not yet sent

Free buffer space where application can write new data to be sent

Old data sent that has already been ACK:ed (Could as well be marked as free space)

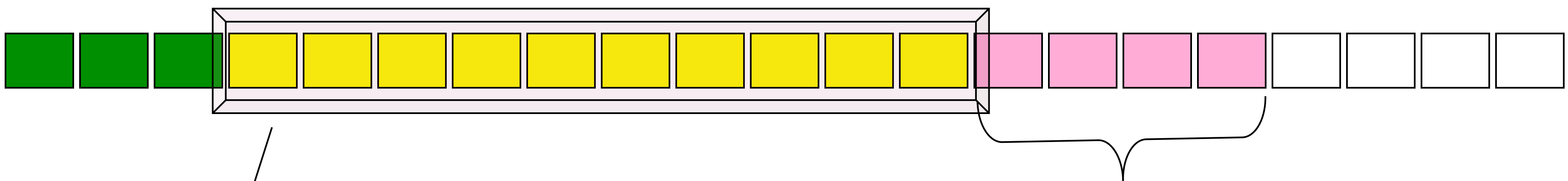
Sliding window - Sender side

Cumulative Acknowledgments

UPPSALA UNIVERSITET

Not sent | Sent, no ACK | ACK:ed | Free

Sending buffer at the sender:



This data can not be sent yet, as the sliding window in this example has a maximum size of 10

Data that has been sent, but not ACK:ed
Also called the *Sending window*

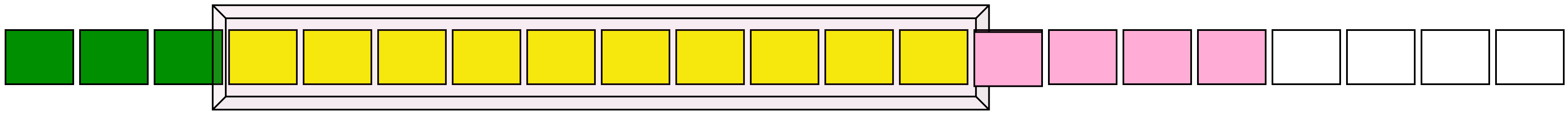
<u>This</u> is the *sliding window* (yes, it slides!)

Sliding window - Sender side

Cumulative Acknowledgments

Not sent   Sent, no ACK   ACK:ed   Free

Sending buffer at the sender:



ACTION: An ACK of the oldest sent packet arrives

- The window *slides* so that the left border is in line with the oldest outstanding ACK

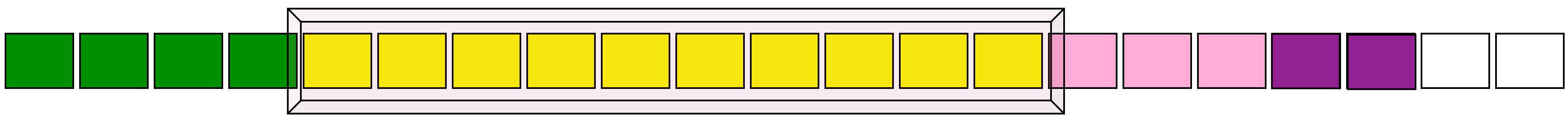
- The unsent segments that fit within the window are sent

Sliding window - Sender side

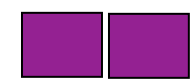
Cumulative Acknowledgments

Not sent | Sent, no ACK | ACK:ed | Free

Sending buffer at the sender:



ACTION: The application has more data to send



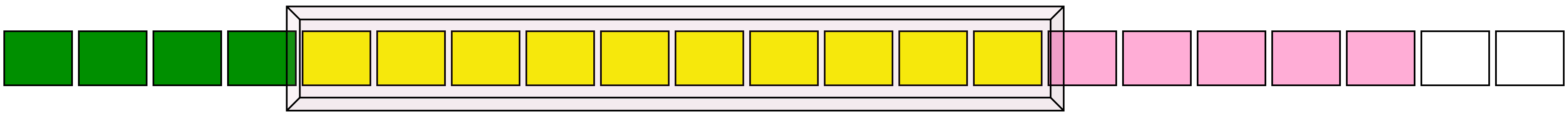
- The data is placed in free buffer slots

Sliding window - Sender side

Cumulative Acknowledgments

Not sent Sent, no ACK ACK:ed Free

Sending buffer at the sender:



ACTION: An ACK arrives in the middle of the window

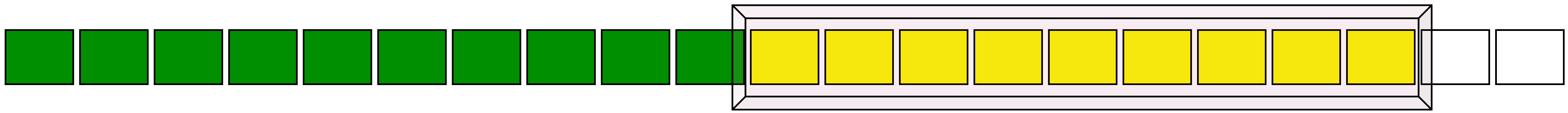
- Older sent but un-ACK:ed segments are now considered to be ACK:ed
- The window slides and unsent segments within the window are sent
- The window shrinks by one segment as there is no more than 9 segments outstanding

Sliding window - Sender side

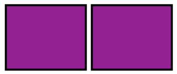
Cumulative Acknowledgments

Not sent Sent, no ACK ACK:ed Free

Sending buffer at the sender:



ACTION: The application has more data to send



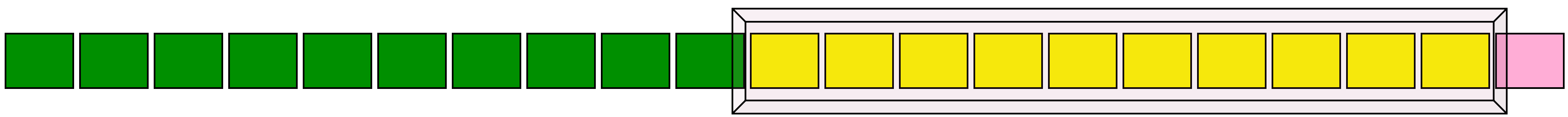
- The data is placed in free buffer slots
- As the window is currently 9 segments wide, it can grow by one segment
- The new data that fits within the window is sent

Sliding window - Sender side

Cumulative Acknowledgments

Not sent | Sent, no ACK | ACK:ed | Free

Sending buffer at the sender:



ACTION: An ACK of already ACK:ed segments arrives

- The ACK is silently ignored

Sliding window - Sender side

Cumulative Acknowledgments

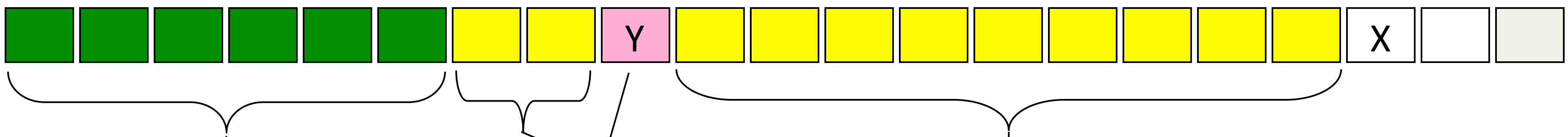
- Must keep track of outstanding data
  – Data sent, but not ACK:ed
- Must not exceed maximum window size
  – Configuration parameter
  – Affects memory consumption
- Must adjust window size when
  – An ACK inside the window arrives
  – New data that can fit within window arrives from application

Sliding window – Receiver side

Cumulative Acknowledgments

Not received | Received | Read | Free | N/A

Read buffer at the receiver:



Data that was previously received, but not yet delivered to the application

Data not yet received

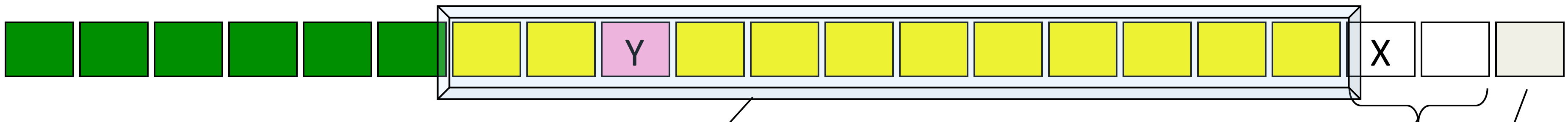
Data that was previously received and that has been delivered to application

Sliding window – Receiver side

Cumulative Acknowledgments

Not received Received Read Free N/A

Read buffer at the receiver:



The sliding window holds data received but not yet read. Must also be able to keep ”holes” like segment Y in the segments

This sliding window has size 12, max size 14

Free buffer space where new segments that are received can be stored

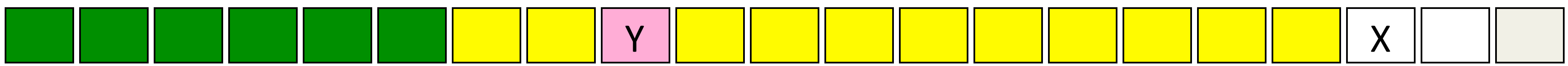
Space unavailable to new segments

Sliding window – Receiver side

Cumulative Acknowledgments

Not received | Received | Read | Free | N/A

Read buffer at the receiver:



ACTION: Segment X arrives

- Store in read buffer, register as received
- Send cumulative ACK Y to indicate that receiver is waiting for Y

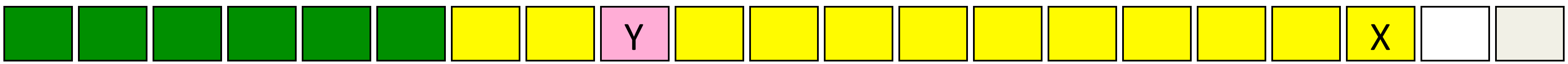
UPPSALA UNIVERSITET

Sliding window – Receiver side

Cumulative Acknowledgments

Not received Received Read Free N/A

Read buffer at the receiver:



ACTION: Segment X+2 arrives

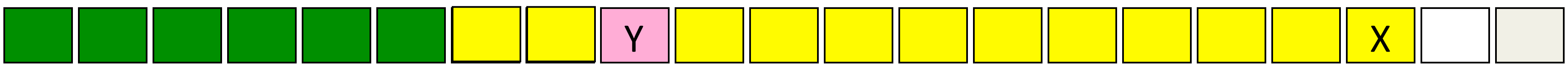
- Can not fit into the buffer, must be discarded
- Send cumulative ACK Y to indicate that receiver is waiting for Y

Sliding window – Receiver side

Cumulative Acknowledgments

Not received | Received | Read | Free | N/A

Read buffer at the receiver:



ACTION: Applications try to read 5 segments

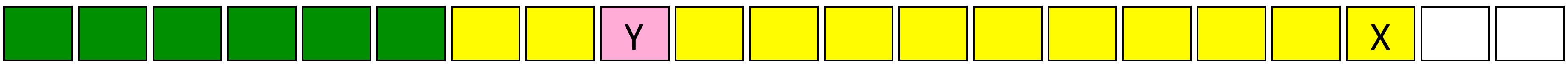
- Only two segments are returned, still waiting for Y
- Application is informed of how much data was read
- The unavailable segment at the end of the buffer becomes available

Sliding window – Receiver side

Cumulative Acknowledgments

Not received | Received | Read | Free | N/A

Read buffer at the receiver:



ACTION: Segment Y arrives

- Store in read buffer, register as received
- Send cumulative ACK (X+1) to indicate that receiver is waiting for (X+1)

X+1

Sliding window – Receiver side

Cumulative Acknowledgments

Role of sliding window is different at receiver

- Represents the maximum buffer size for segments received but still not read
- If a segment that does not fit inside the window arrives (either too new or too old), it is discarded.
  - However, an ACK is sent
- To avoid running out of buffer space, receiver can inform sender about available buffer space in each ACK

Damaged messages

- Perform integrity check
  - Checksum, CRC etc
  - Parity bits
  - …
- Which one to use?
- Should an ACK be sent or not?

Proactive actions

- A lost message means *something*
  - Congestion in the network
  - Congestion at the receiver
  - Interference
  - …
- A retransmission means *something* too
  - Too short timeout timer at sender
  - Lost ACK (see above)
  - Possibly network problems
  - …
- Can these signals be used somehow?

Reactive actions

- Assumption: Network congestion
  - Reduce packet rate
- Assumption: Receiver congestion
  - Wait for receiver to catch up
- Assumption: Large delay variations
  - Adjust timeout timer

UPPSALA UNIVERSITET

General principles in networks

- "Be liberal in what you receive and conservative about what you send"
  - Make use of as much information as possible
  - Do not send data likely to be lost anyway

- End-to-end principle
  - Expect nothing from intermediate networks
  - Required functionality in end hosts

- Smart senders, dumb receivers
  - Follow a base rule set to be compatible
  - Can use different approaches to reliability

Possible aids for proactivity

- Cumulative acknowledgments
- Acknowledgment vectors
- Signaling mechanisms in the network

How to react on different signals

- Unexpected ACK
  - Traffic is flowing, but something has happened
    - Reordering?
    - Single packet loss? Means what?

- Timeout
  - Severe network error, especially if using some sort of pipelining

What is being used in the Internet?

- Transport layer: TCP
  - Cumulative acknowledgments
  - Congestion control
  - Congestion avoidance
  - Flow control
  - Internet checksum
- Network layer:
  - Internet checksum
  - Limited feedback
- Link layer:
  - Depends on access network technology

How can all equipment involved get along?

- There are rules, *protocols*
- A protocol defines:
  - What different messages look like
  - What happens when a message is received
- Each protocol has a limited task
  - Sending a message over a link
  - Sending messages in the right direction
  - Requesting a web page from a web server
  - …

How can my message be delivered to the right destination?

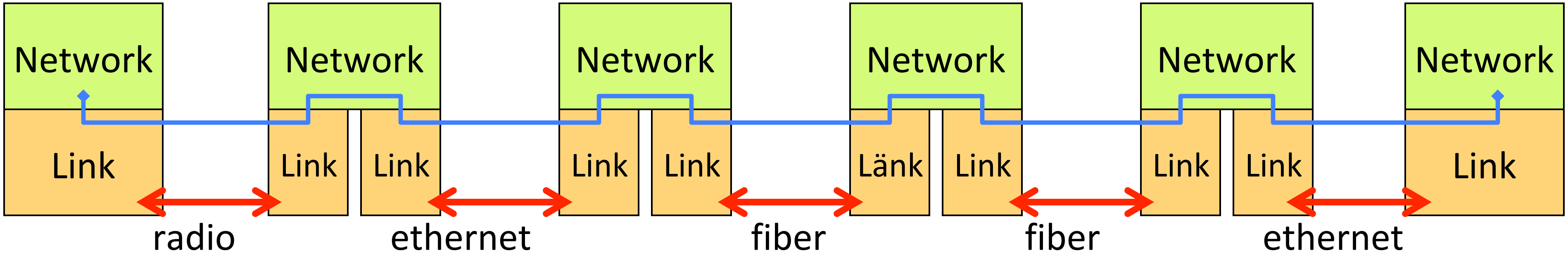
- Many intermediate nodes
  – At each node, a decision about how to forward your message is made

- Identifying sender and receiver
  – All intermediate nodes must use a common *naming* convention

- What happens if something goes wrong?
  – Can be dealt with locally or *end-to-end*

Principal overview

- Communication occurs in several *layers*
  - The *link layer* forwards a message to another node connected to the same physical media
  - The *network layer* makes decision about onto which link a message shall be forwarded
    - Same type of identifier is used in all nodes



Resolving names

- Behind every link, there is an address
  - Sometimes entered manually
  - Example: http://www.it.uu.se/edu/
    - http — protocol to be used
    - www.it.uu.se — server to contact
    - /edu/ — web page to request

- How does the web browser know where to send the request?

Name translation

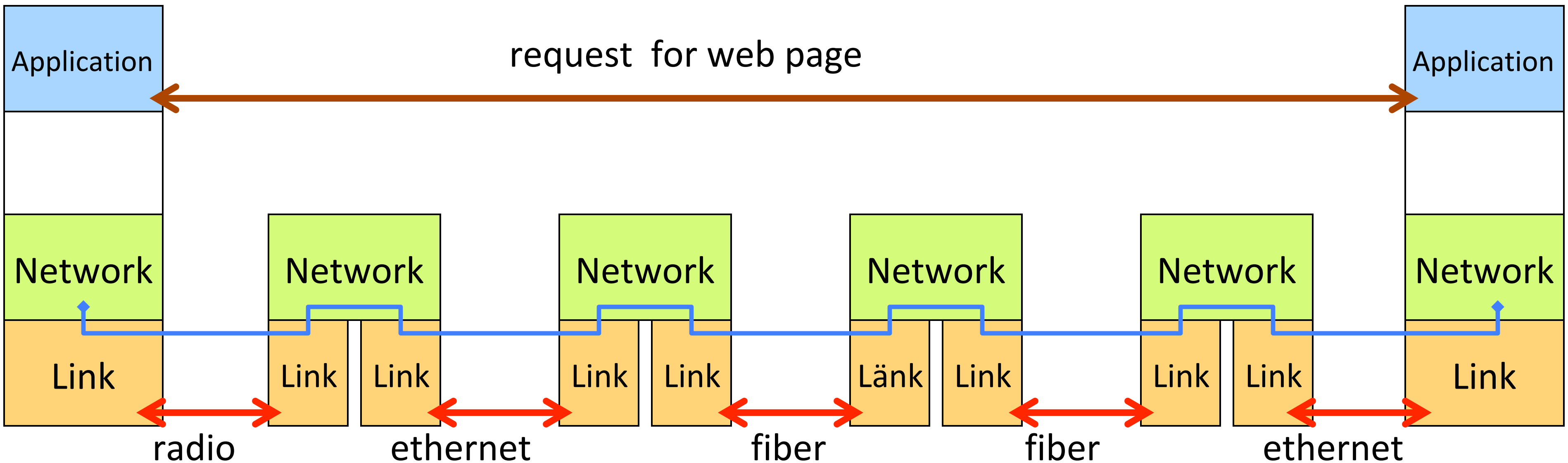
- Different naming schemes at different levels
  - Applications use application-specific schemes
  - The network uses something else
  - Different link technologies use different schemes

- It must be possible to translate between different types of addresses
  - For now, let us just assume that is possible

What happens in the web browser when I click on a link?

- The server address is identified (www.it.uu.se)
- The server address is translated into a network address (130.238.12.100)
- A http request is sent to the host with that network address

Extended principal overview



What is missing?

- How does the network layer work?
  - What does addresses look like?
  - How are they parsed?
  - How are messages forwarded?

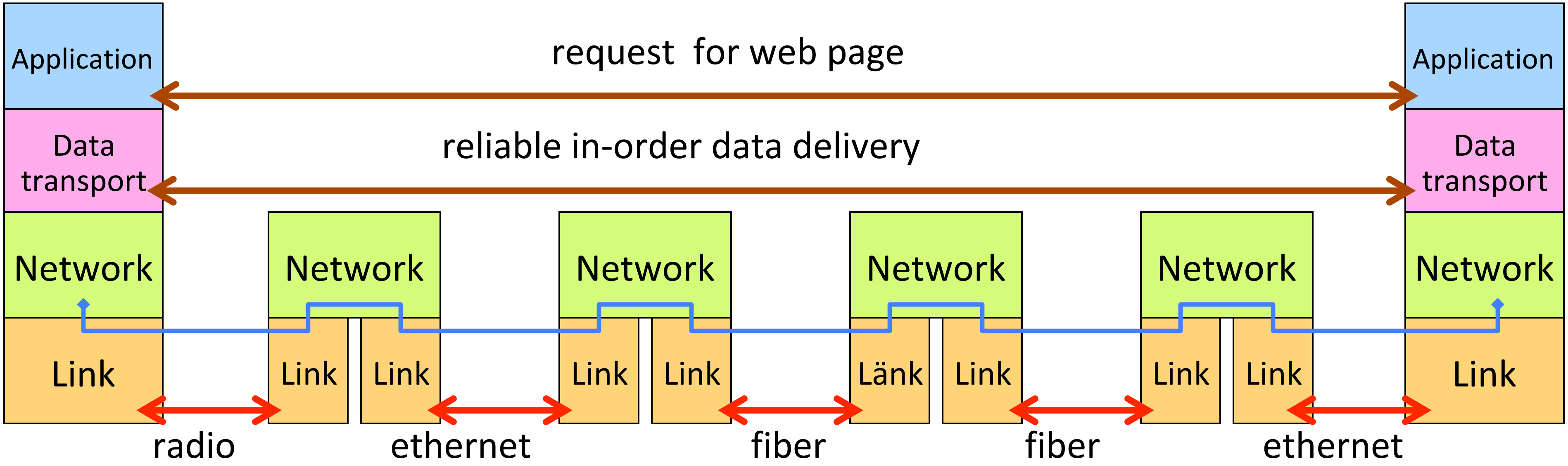
Introducing: *Internet protocol (IP)*

- The protocol in the network layer
  - Global delivery of messages
    - using IP addresses
  - No guarantees
    - Messages can be reordered
    - Messages can experience different delays
    - Messages can be broken when received
    - Messages can arrive out-of-order
  - "Best-effort delivery"

Is “best-effort” good enough?

- No!
  - We want all messages to arrive in the correct order without being broken
  - Something is needed between the network layer and the applications to deal with this
    - Uses the best-effort service of the network layer
    - Enhances it to provide useful services to applicatons
      - Reliability
      - In-order delivery
      - Retransmissions when needed
      - …

Extended principal overview (2)



Different layers of communication

- Application/Services
  - Own addresses
    - www.it.uu.se
    - lln@it.uu.se
  - Exchange *messages*

- Data transport services for applications
  - Different protocols with different features
  - Exchange *segments* or *datagrams*

- Global best-effort message delivery
  - Uses IP addresses
    - 130.238.12.100 (IPv4)
  - Exchange *packets*

- Message delivery between two nodes attached to the same link
  - Different types of addresses
  - No delivery guarantees
  - Exchange *frames*

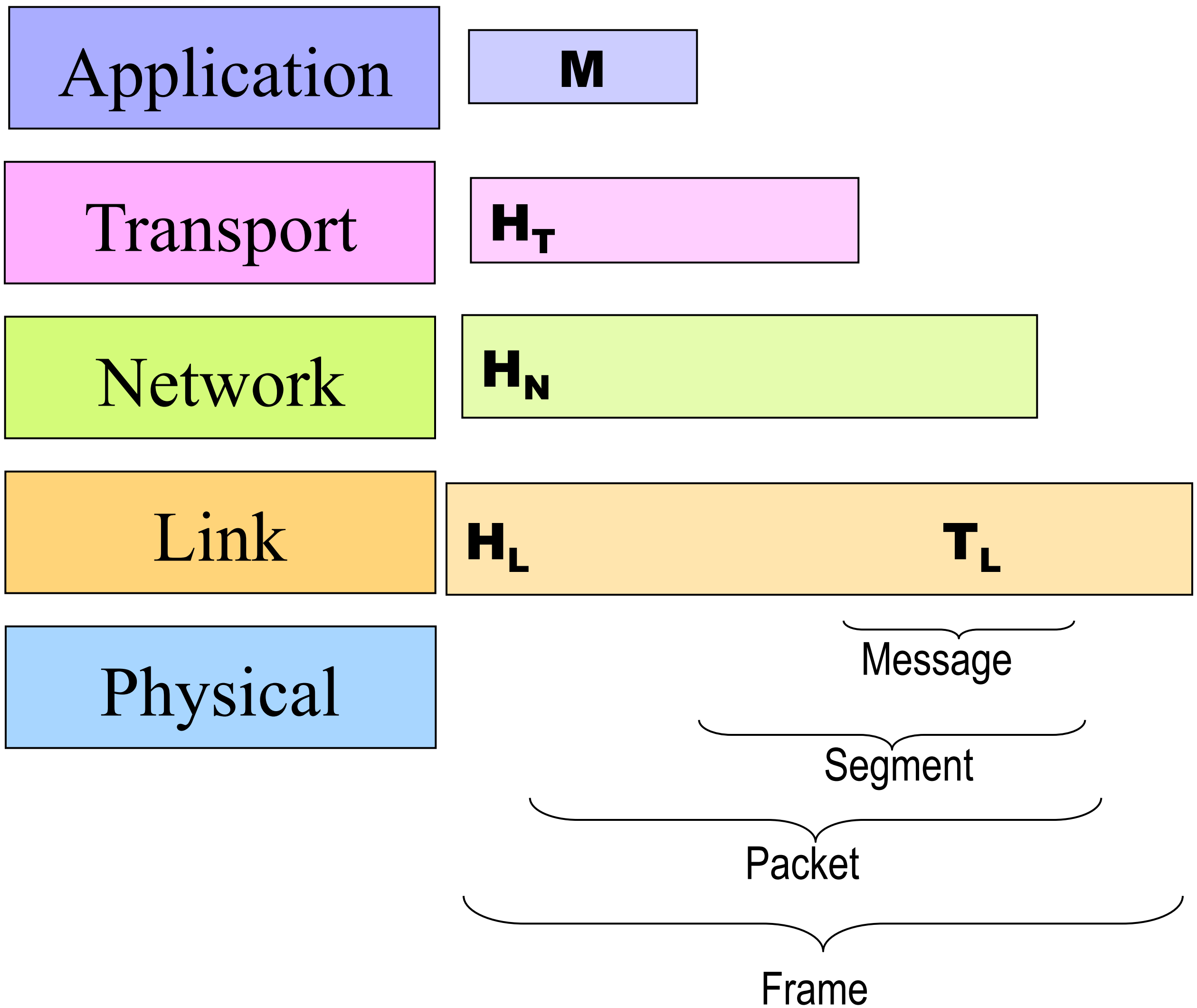
Application layer

Transport layer

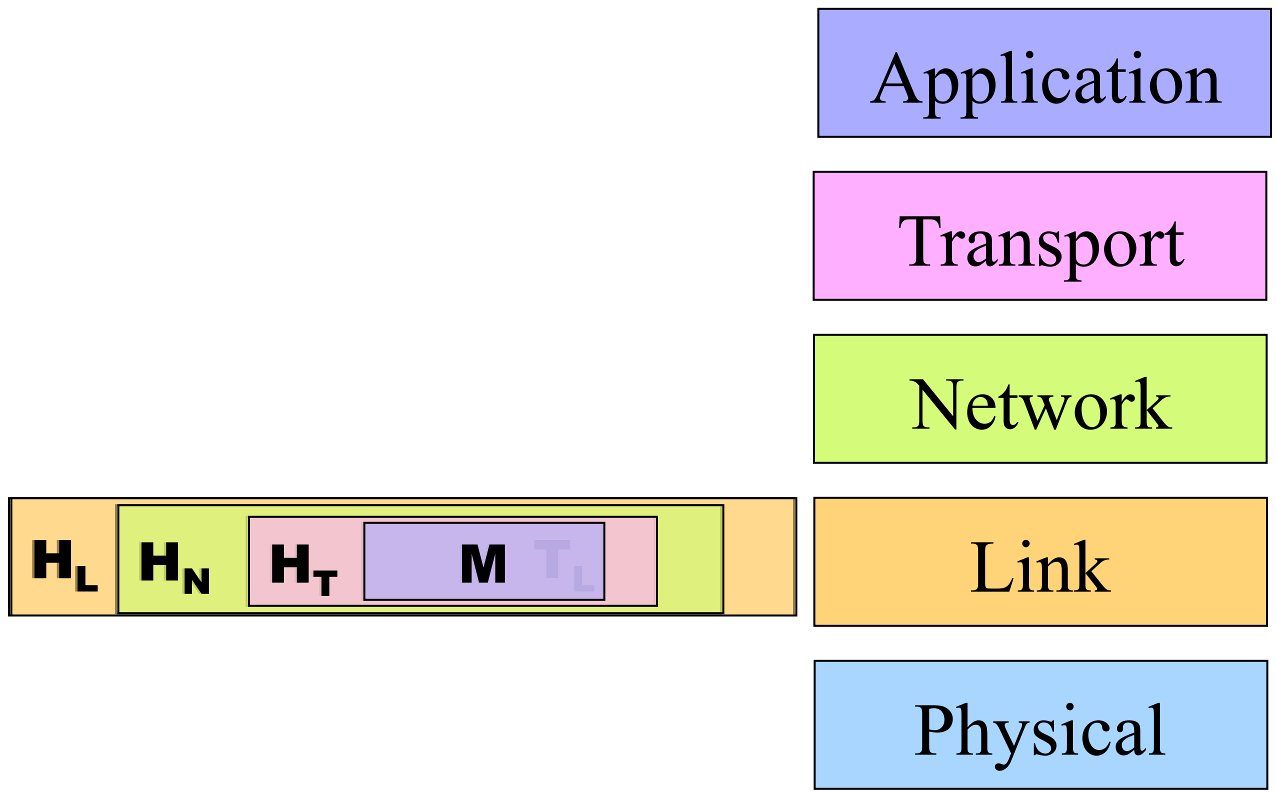
Network layer

Link layer

How layering works



How layering works (2)



UPPSALA UNIVERSITET

Summary

- The Internet architecture is based on **Statistical multiplexing**
- Data are sent in **packet** using a **best-effort service**
- Data is moved between **queues** until reaching the destination
- Data is usually lost in the network due to **overfull queues**
- Reliability can be achieved using an **ARQ** scheme
- Data can be pipelined by usage of **sliding windows**
- The Internet architecture is organized in a **4-layer stack**