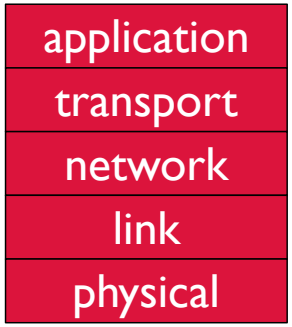
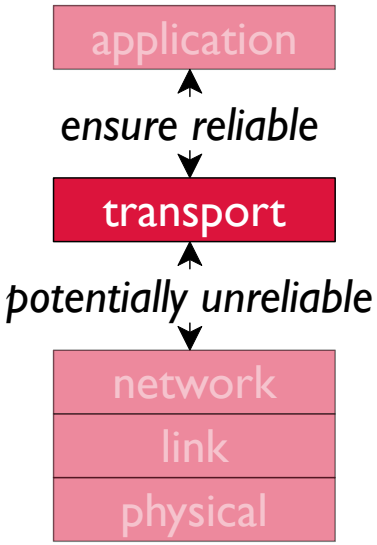
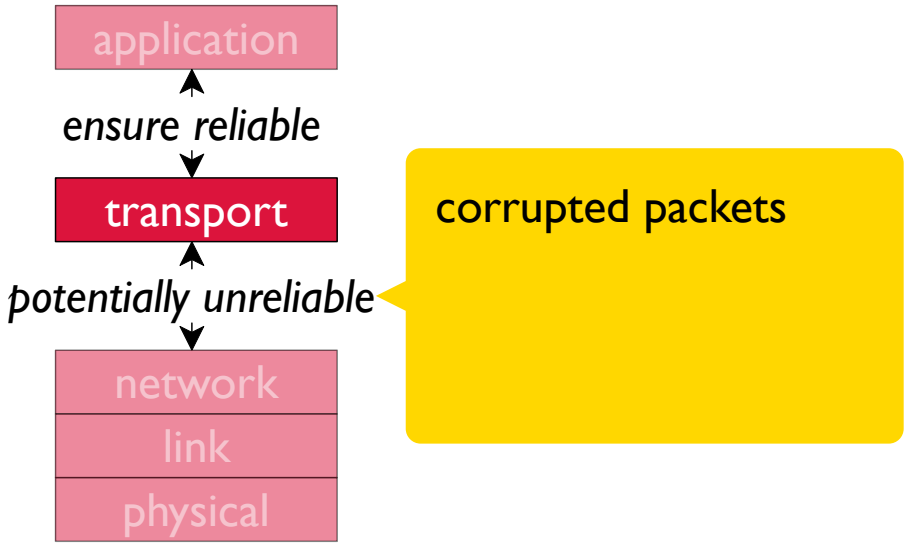
Reliable Data Transfer



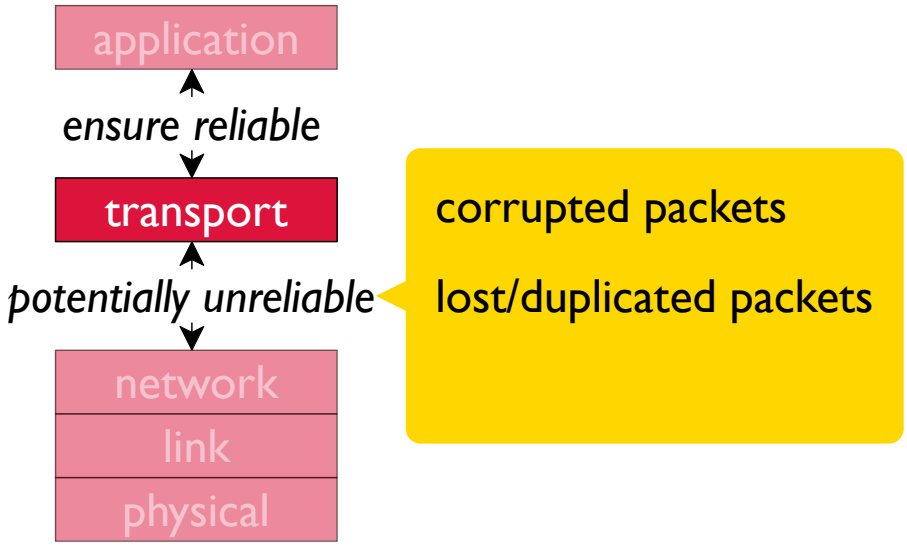
Reliable Data Transfer



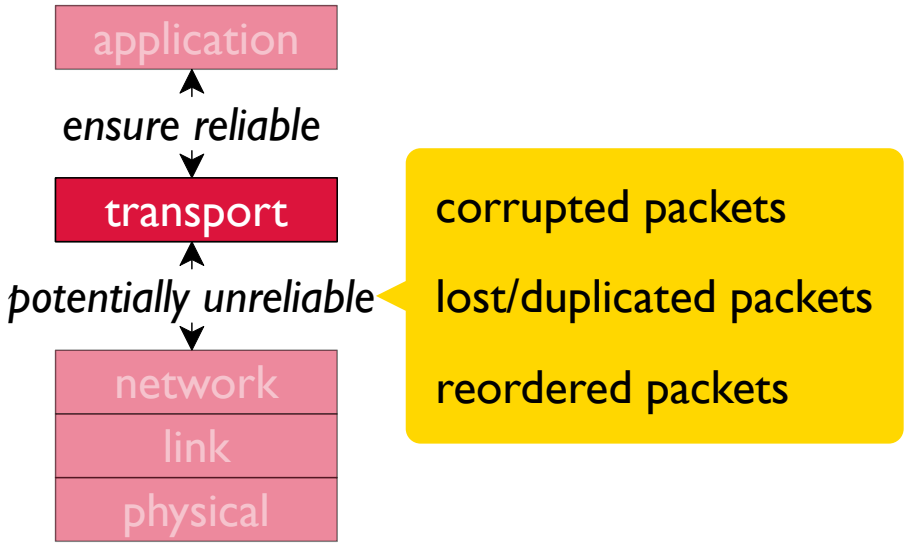
Reliable Data Transfer



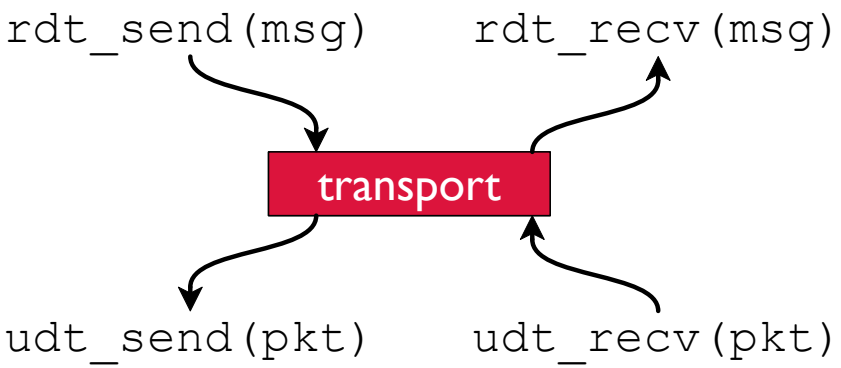
Reliable Data Transfer



Reliable Data Transfer

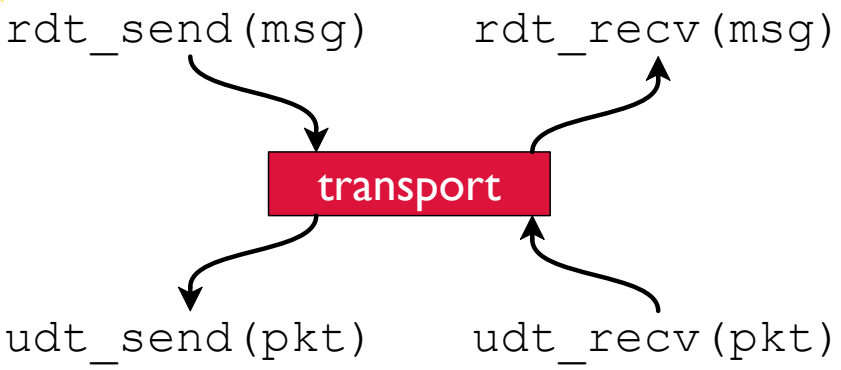


Reliable Data Transfer



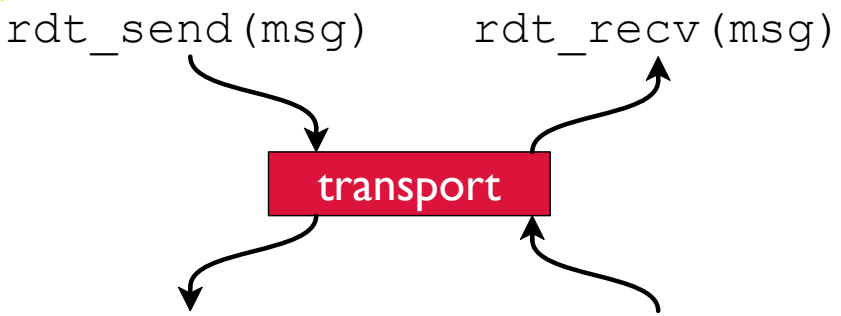
Reliable Data Transfer

"reliable data transfer"



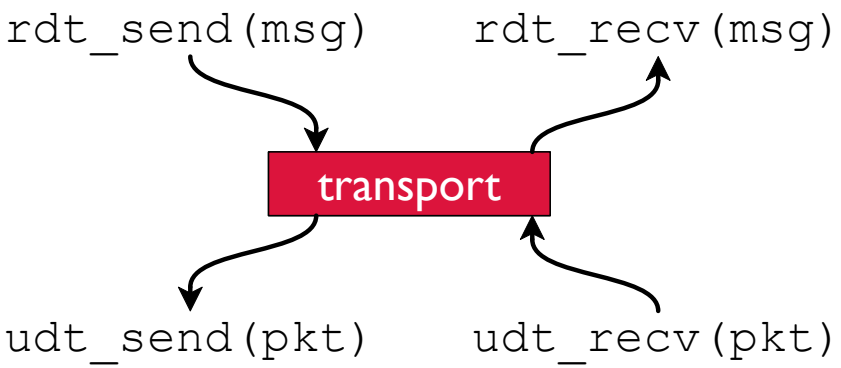
Reliable Data Transfer

"reliable data transfer"

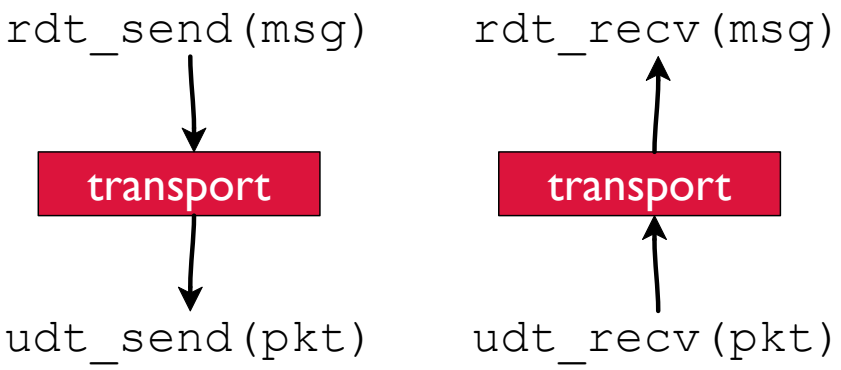


"unreliable data transfer"

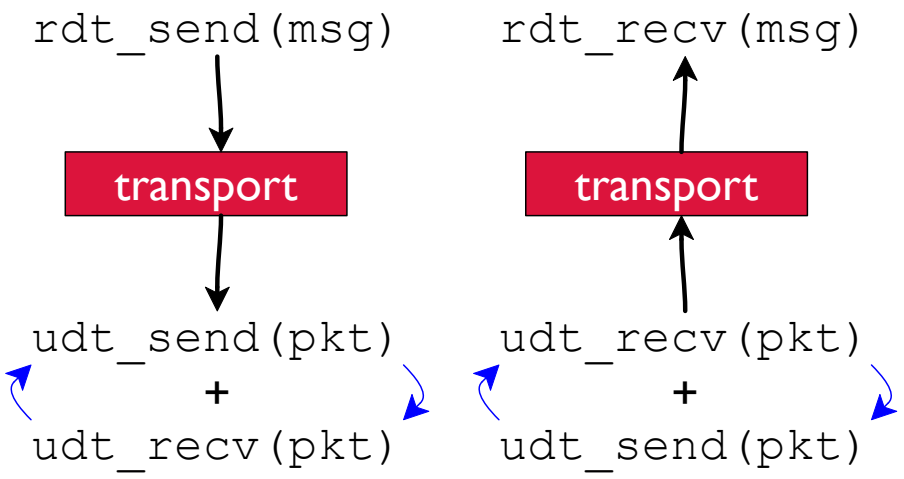
Reliable Data Transfer



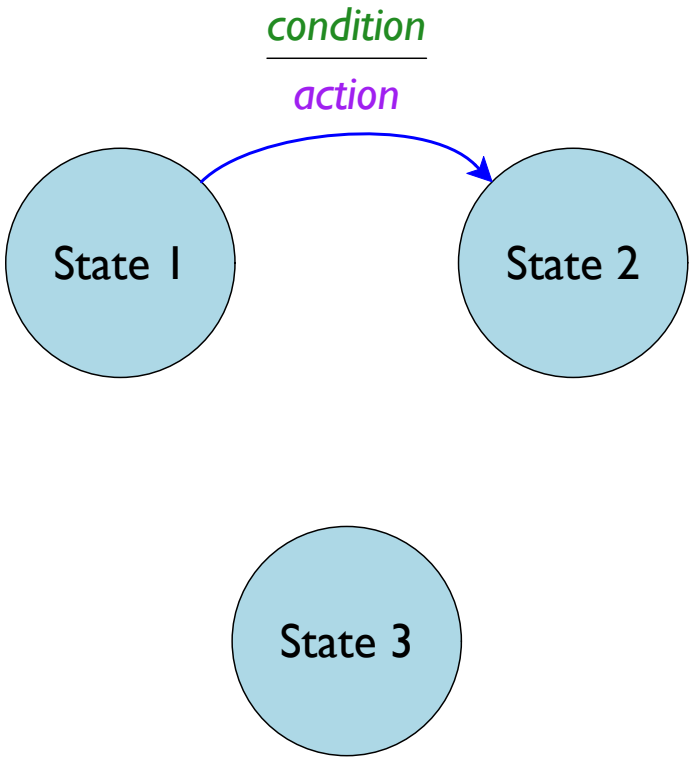
Reliable Data Transfer



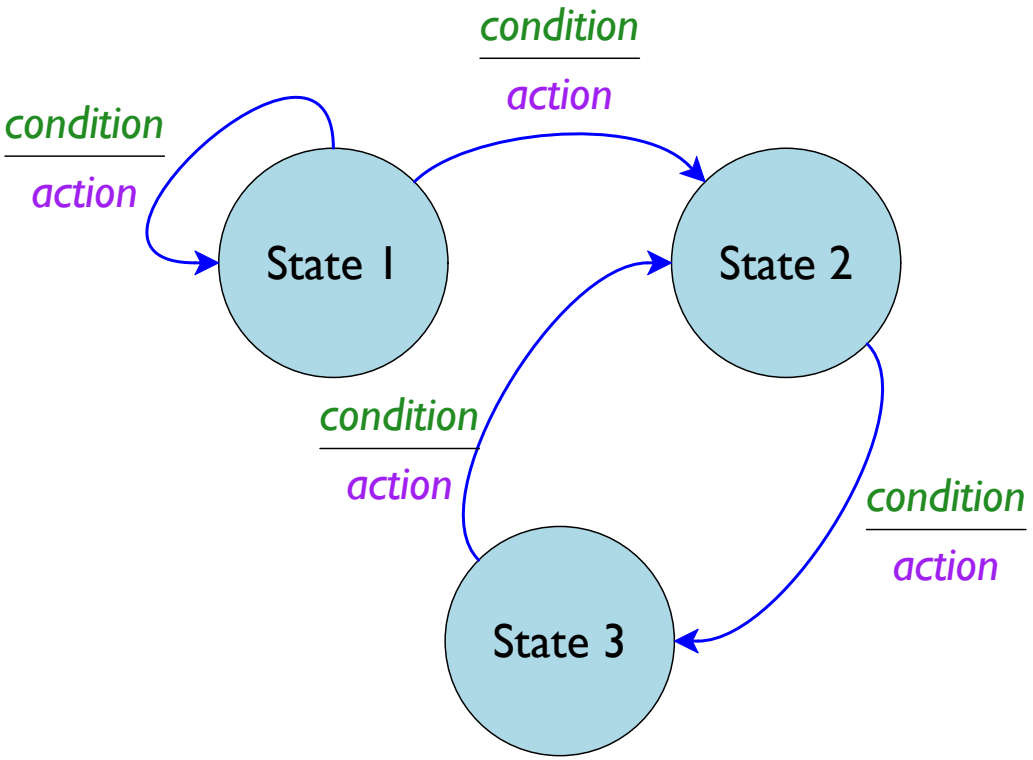
Reliable Data Transfer



State Machines

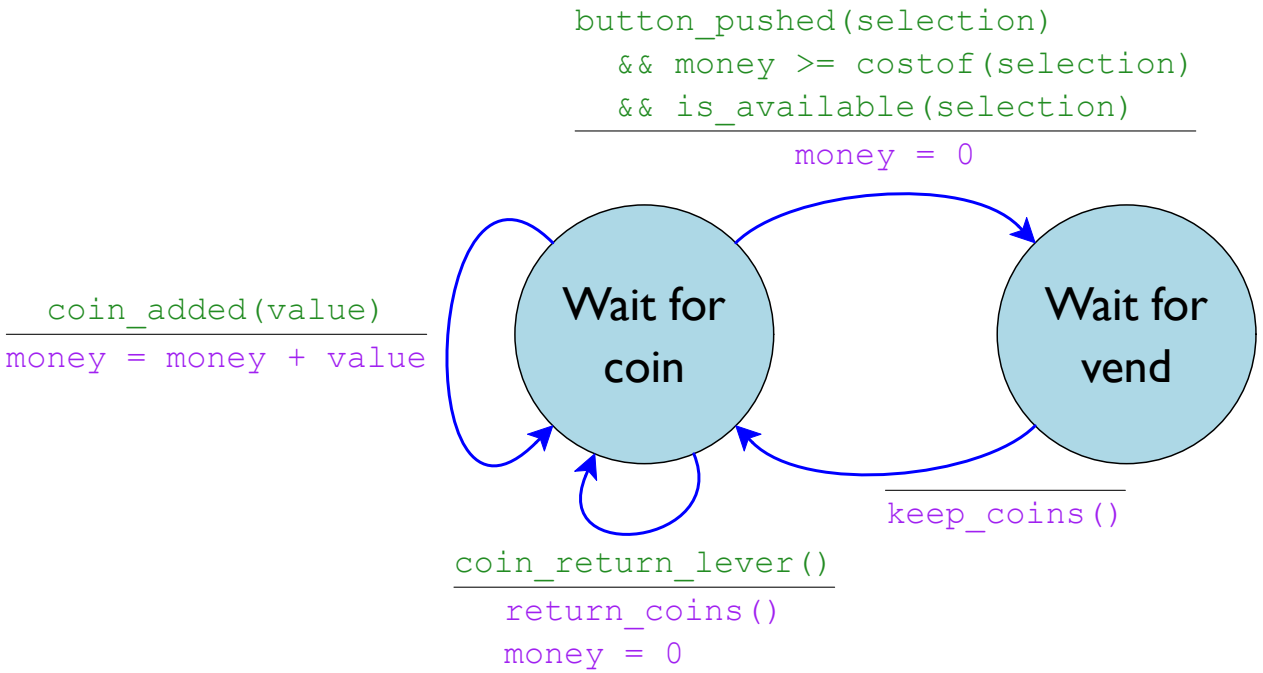


State Machines



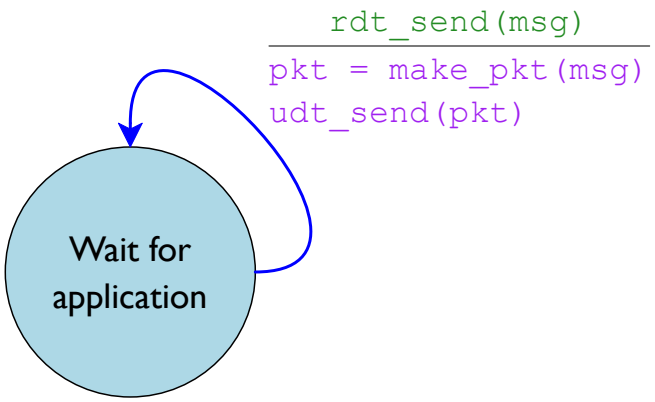
State Machines

Vending machine example:



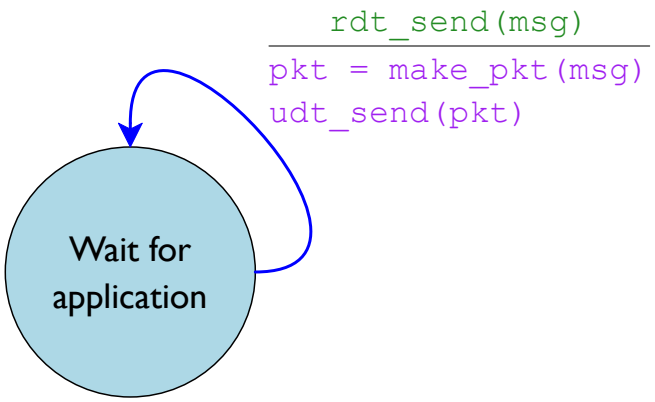
Assuming Reliable `udt_send` and `udt_recv`

sending host

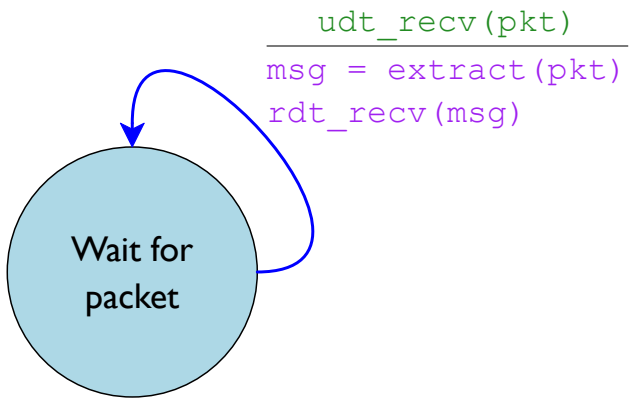


Assuming Reliable `udt_send` and `udt_recv`

sending host



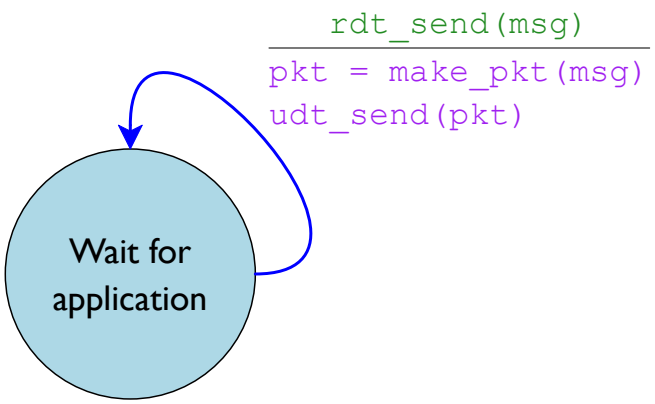
receiving host



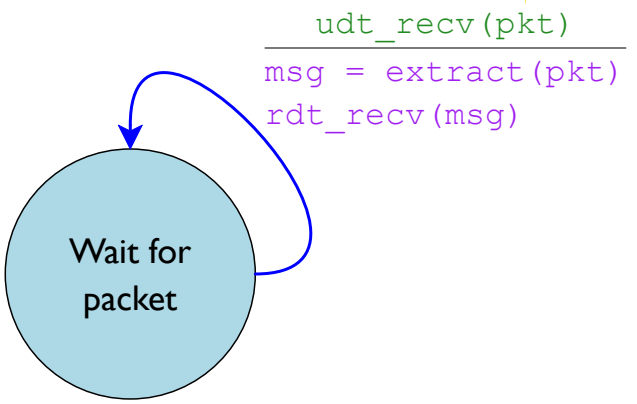
Assuming Reliable `udt_send` and `udt_recv`

What if `pkt` is corrupted?

sending host

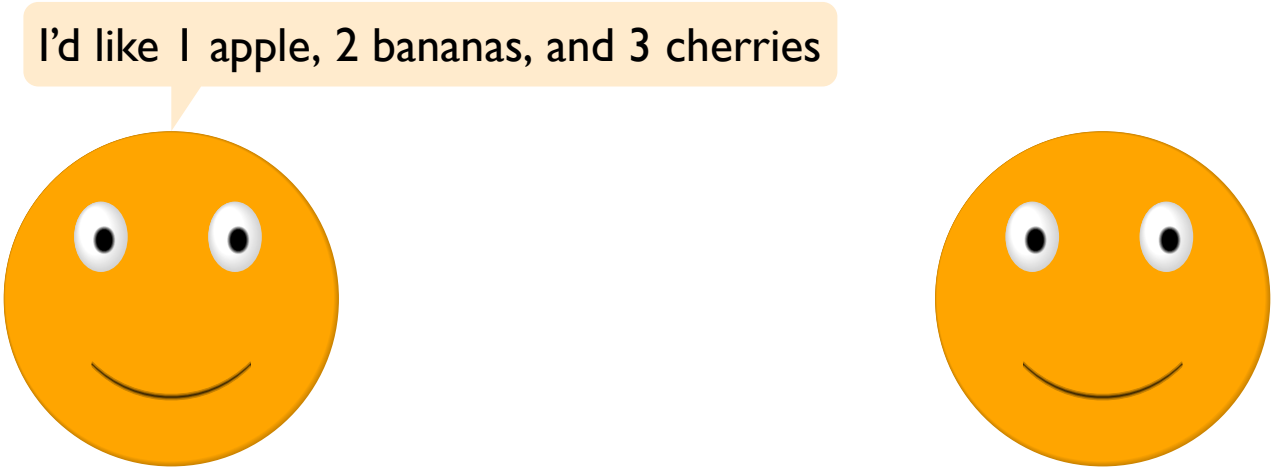


receiving host



Checksum

Checksum

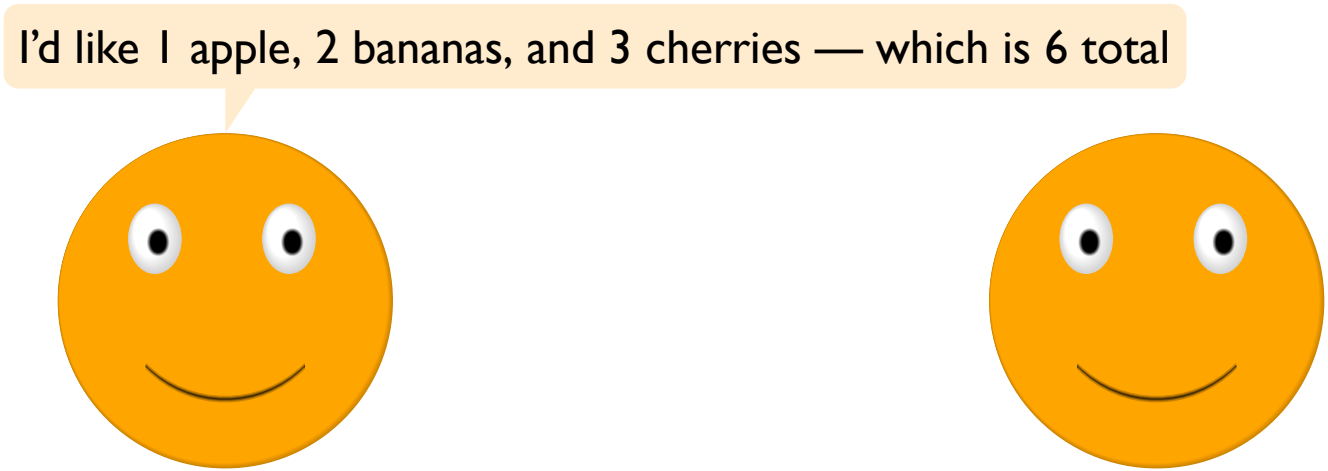


Checksum

I'd like 1 apple, 2 bananas, and 3 cherries

Ok: 1 apple, 2 bananas, and 2 cherries

Checksum



Checksum

I'd like 1 apple, 2 bananas, and 3 cherries — which is 6 total

Ok: 1 apple, 2 bananas, and 2 cherries — which is 6... oops

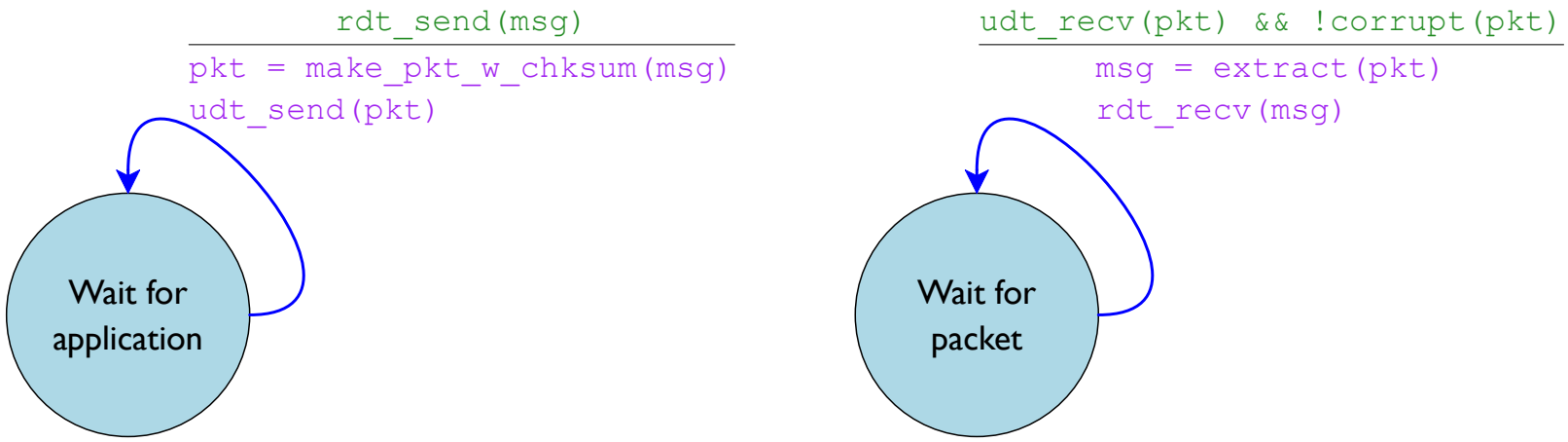
Checksum

I'd like 1 apple, 2 bananas, and 3 cherries — which is 6 total

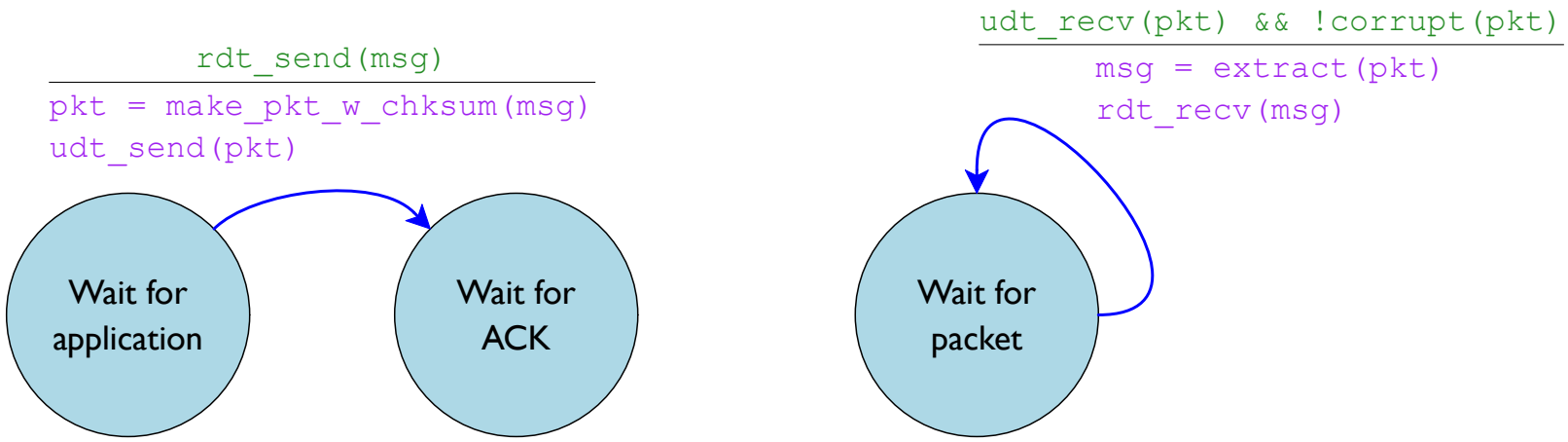
Ok: 1 apple, 2 bananas, and 2 cherries — which is 6... oops

To deal with lots of numbers, just keep low bits of the sum

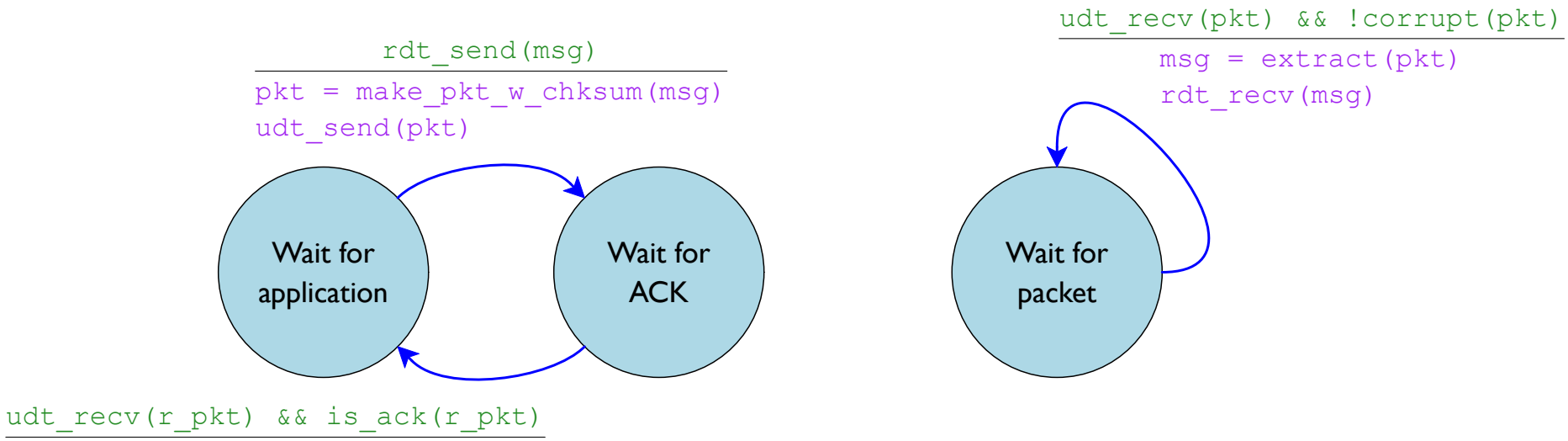
Using a Checksum



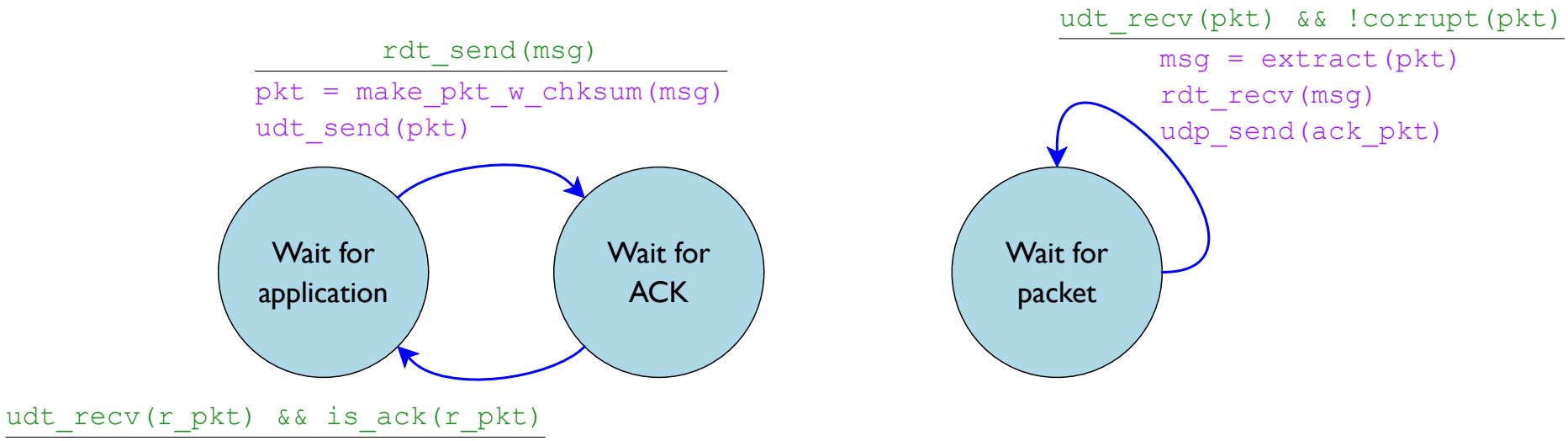
Using a Checksum



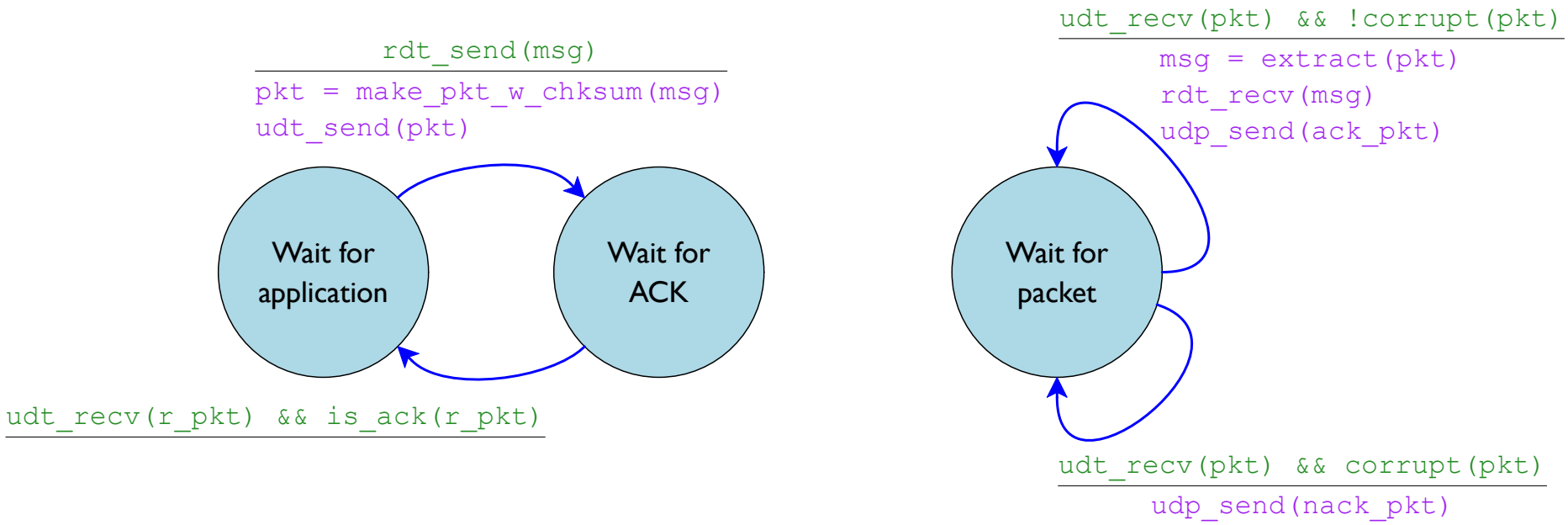
Using a Checksum



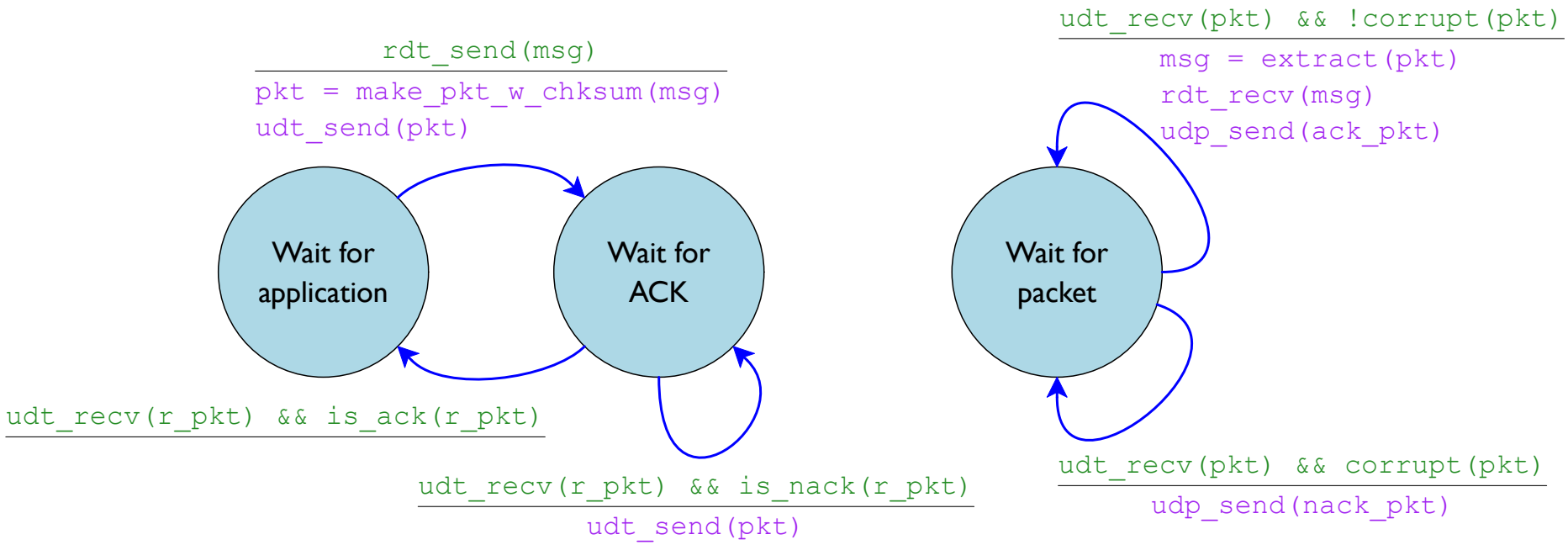
Using a Checksum



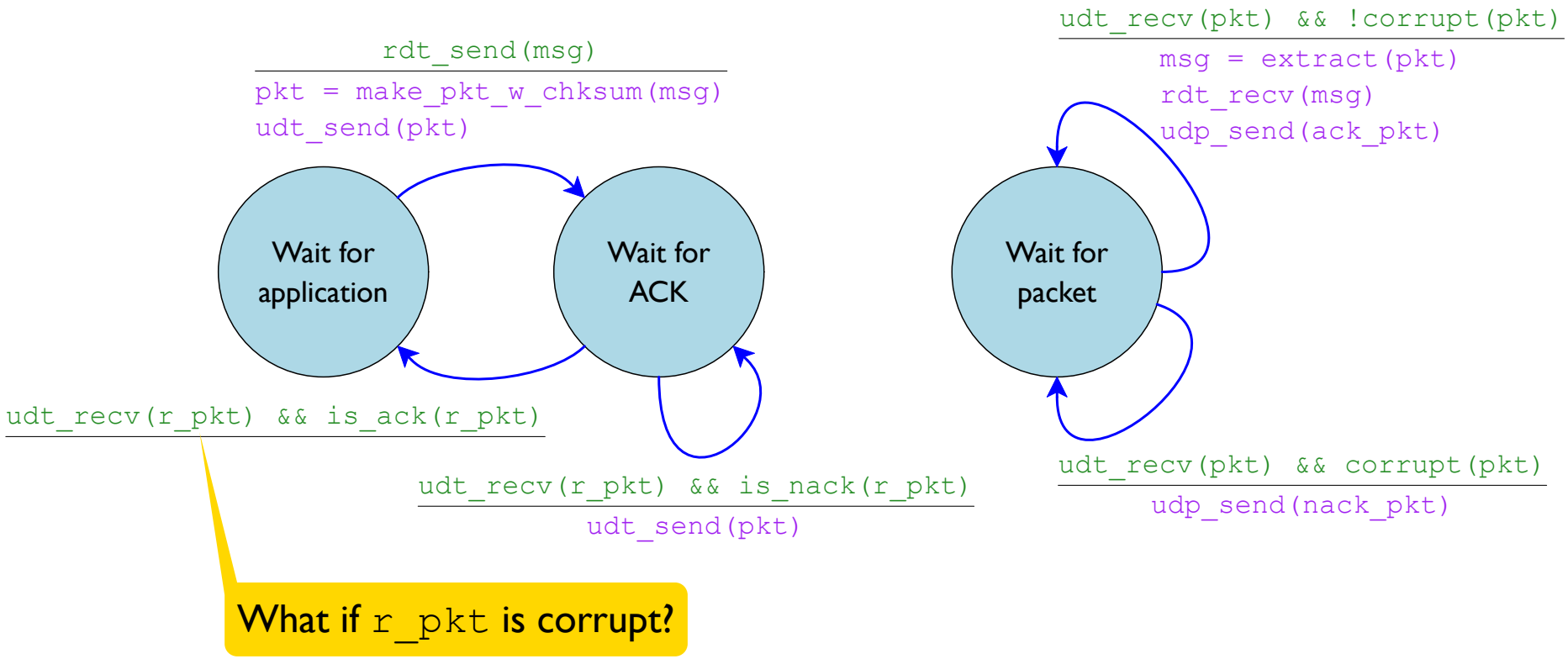
Using a Checksum



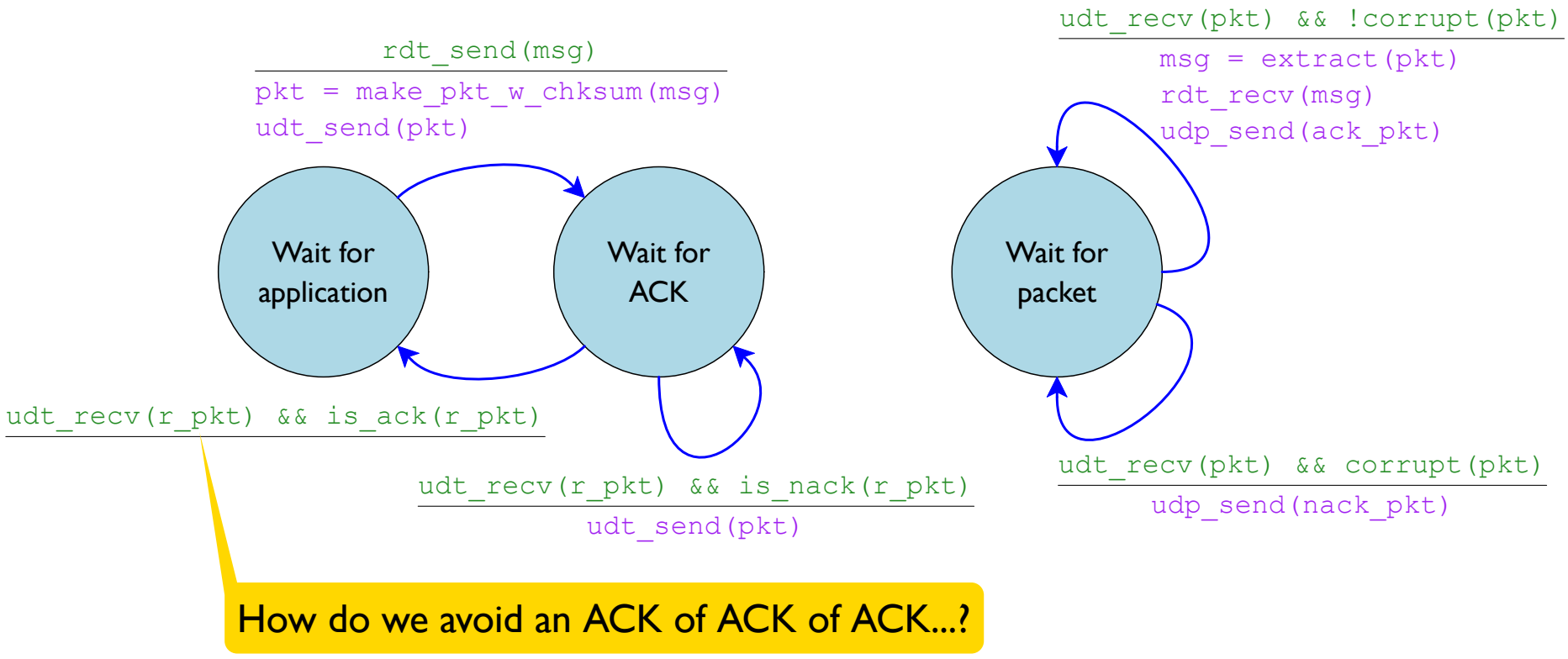
Using a Checksum



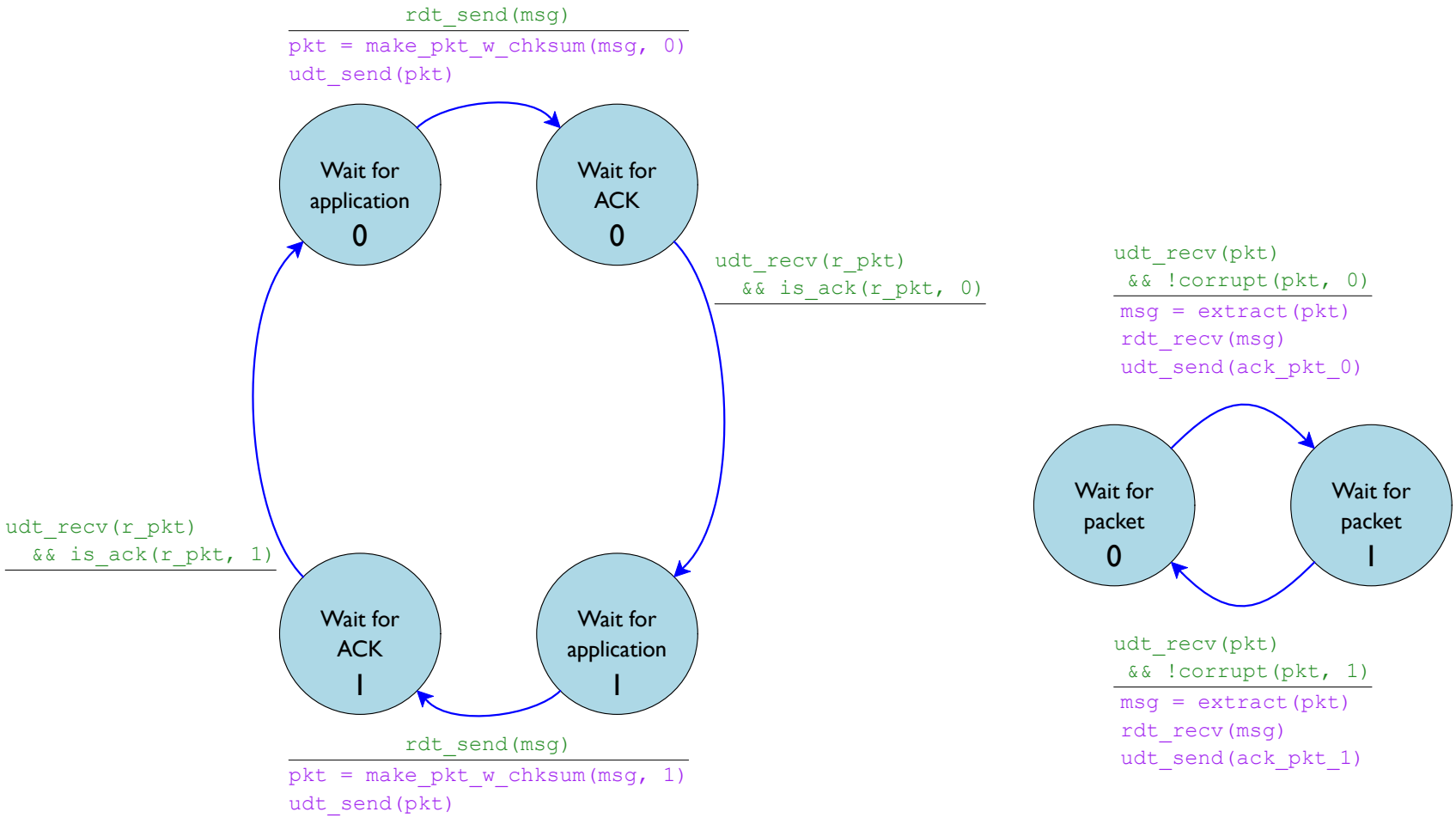
Using a Checksum



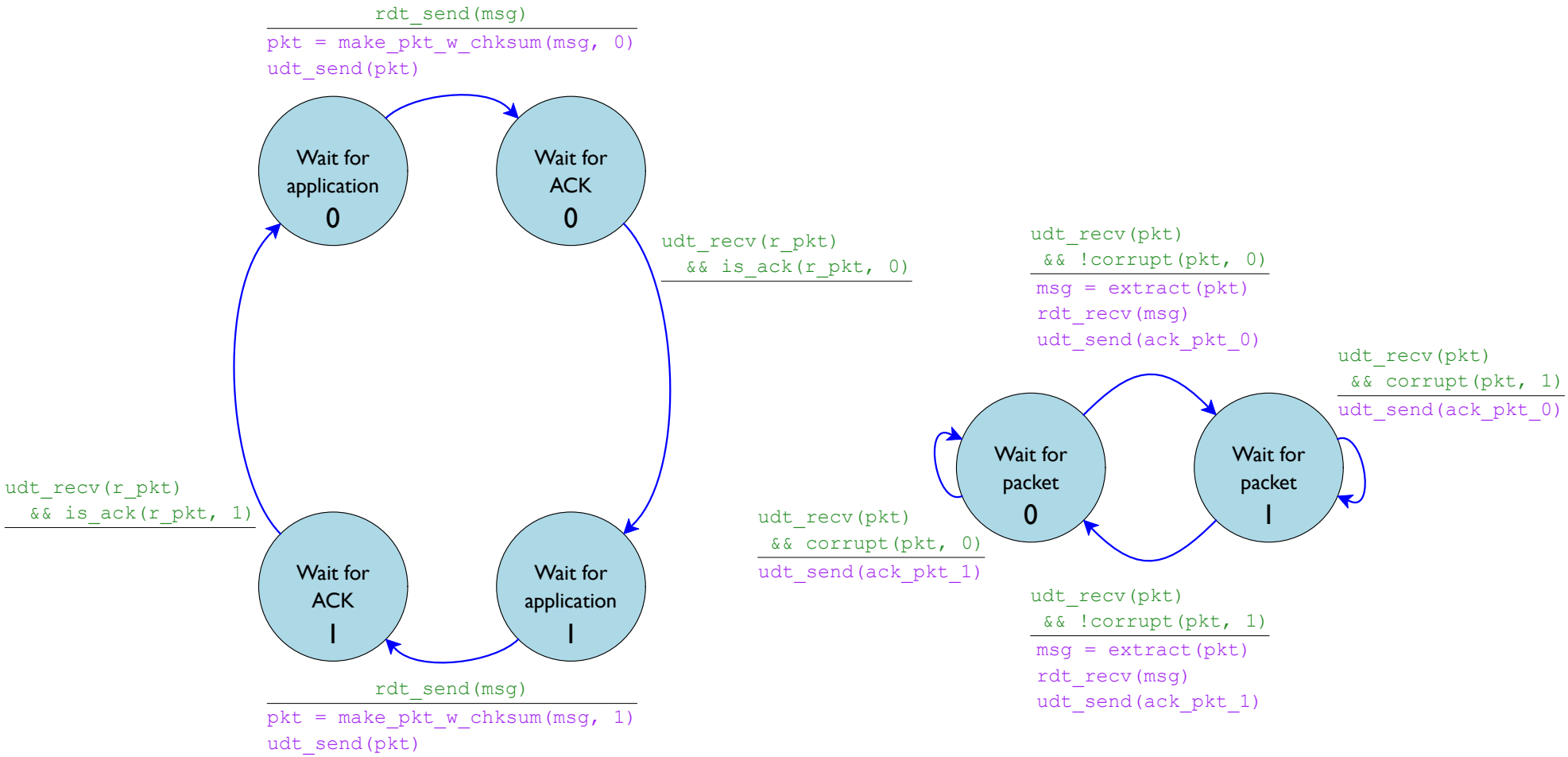
Using a Checksum



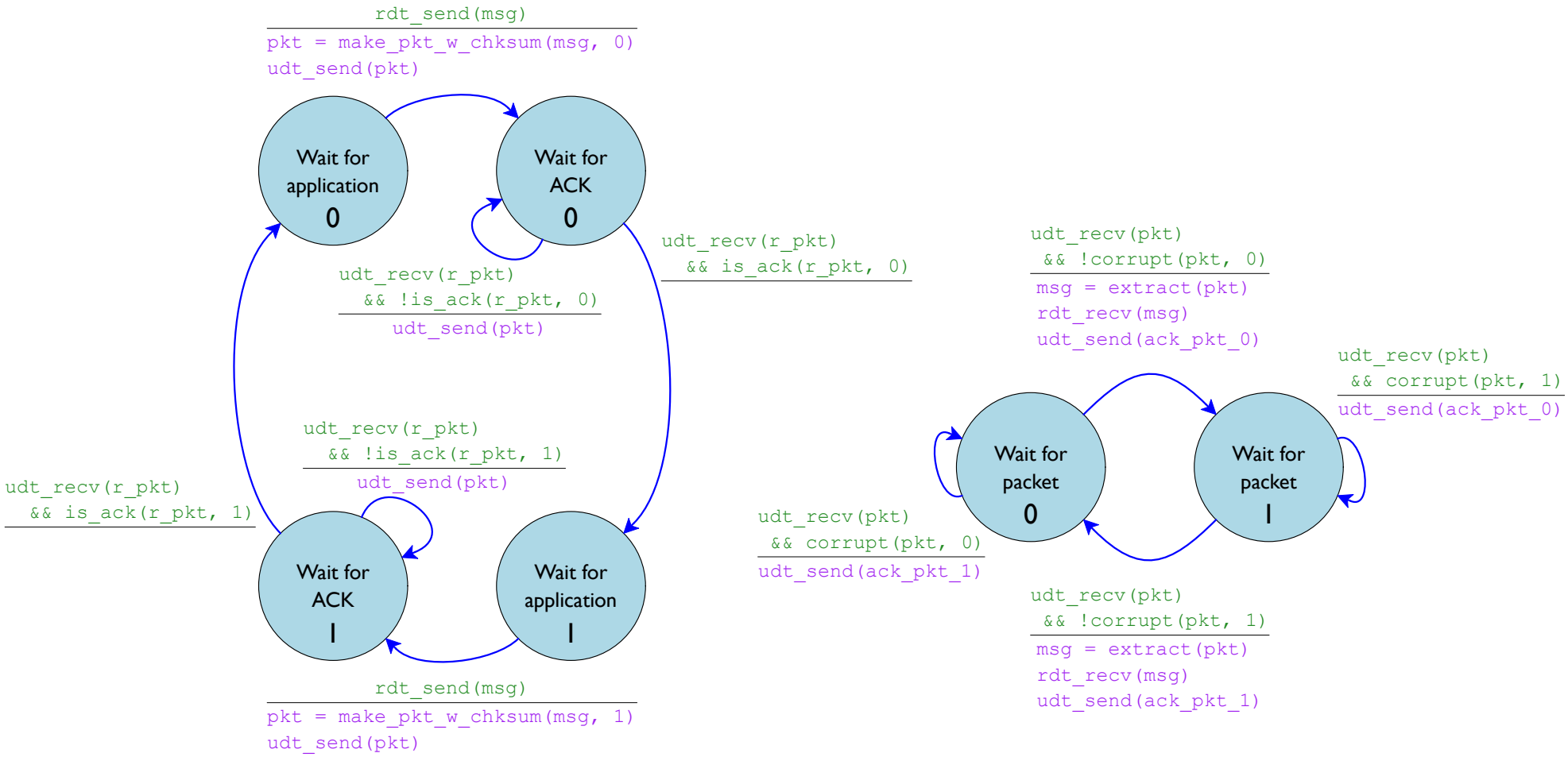
Handling ACK Corruption



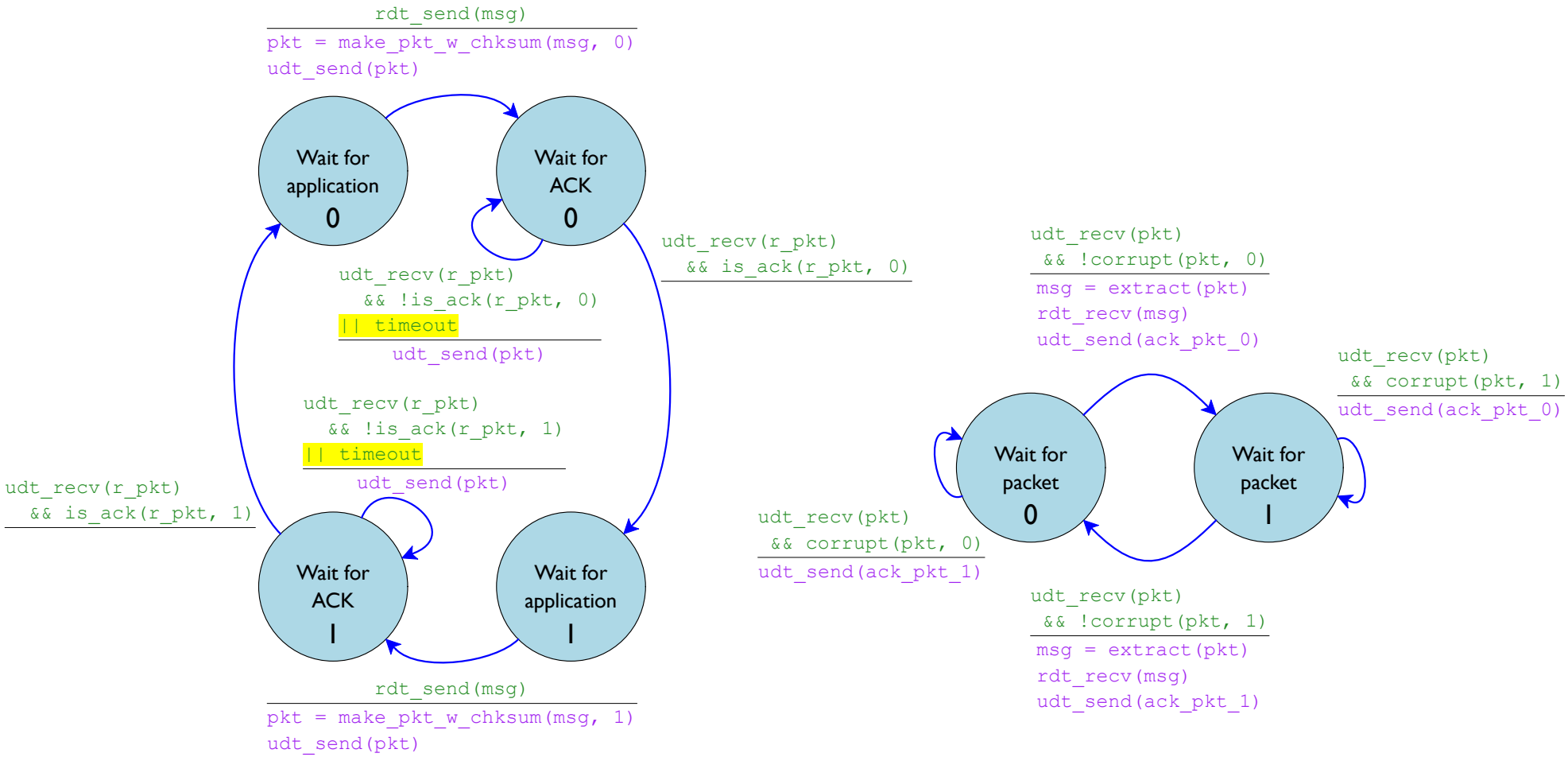
Handling ACK Corruption



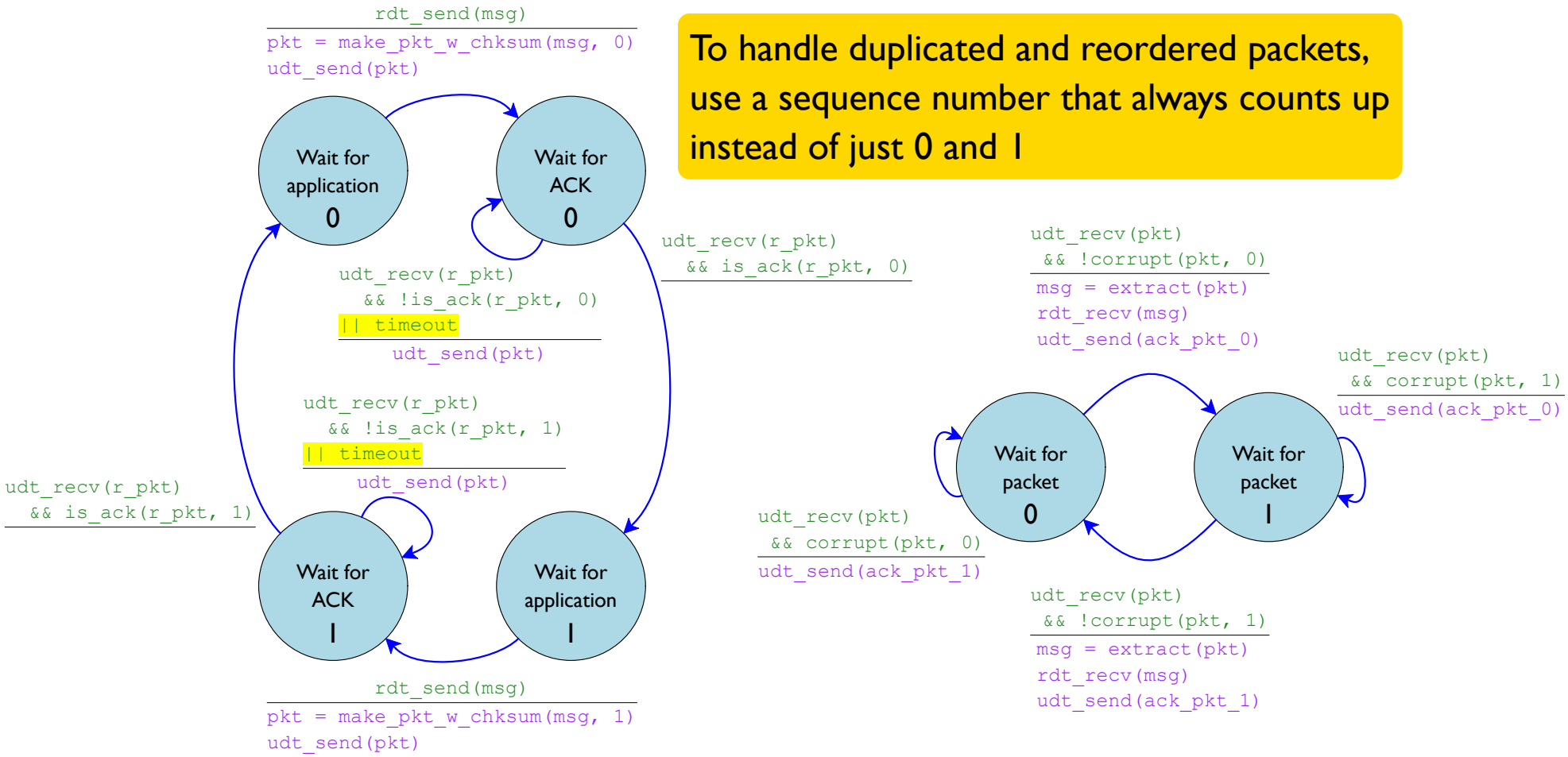
Handling ACK Corruption



Handling Corrupt and Lost Packets



Handling Corrupt and Lost Packets



To handle duplicated and reordered packets, use a sequence number that always counts up instead of just 0 and 1

Choosing a Timeout

RTT is minimum useful timeout

Choosing a Timeout

RTT is minimum useful timeout

- too small ⇒ resend data and ACKs unneceesarily
- too large ⇒ sender waits too long to resend

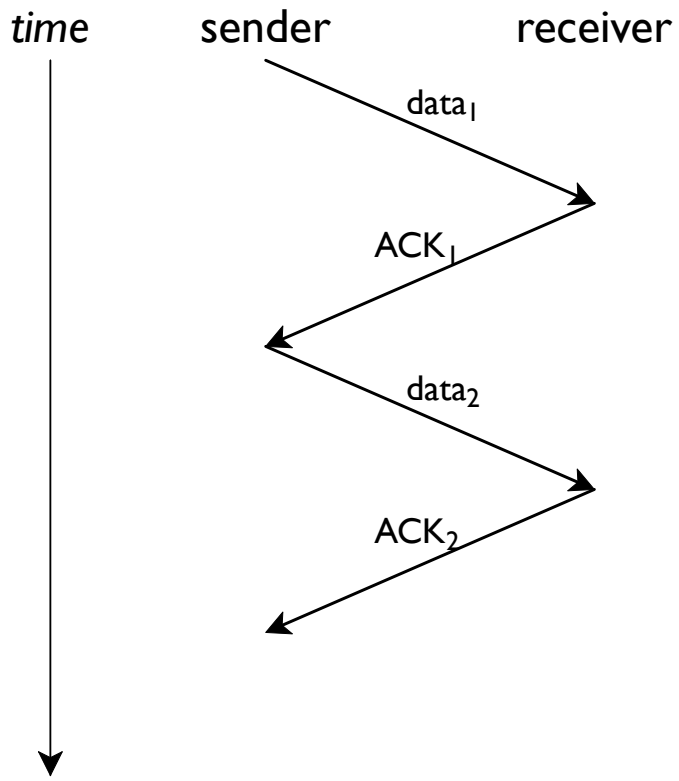
Choosing a Timeout

RTT is minimum useful timeout

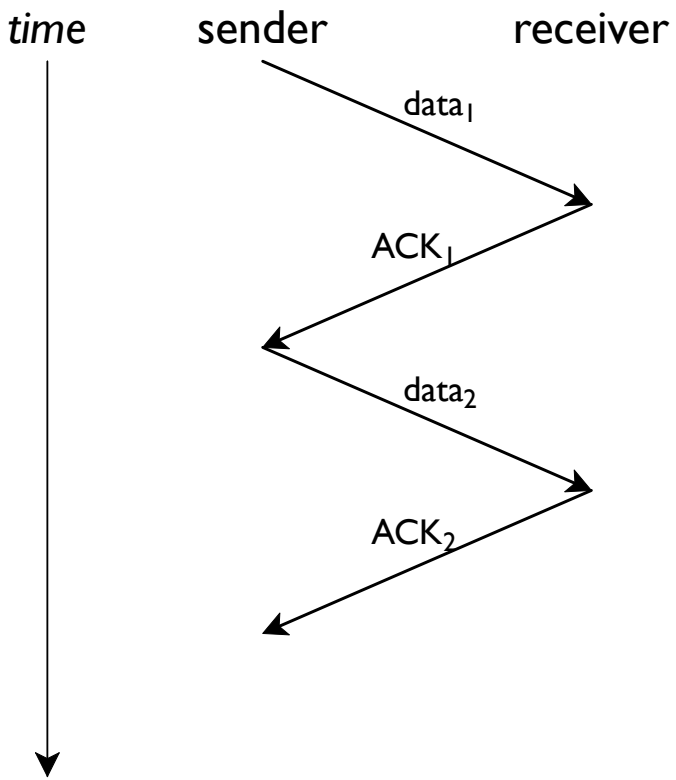
- too small ⇒ resend data and ACKs unneceesarily
- too large ⇒ sender waits too long to resend

scale × avg(RTT) + stddev(RTT) is a good approach

Sequential Messages

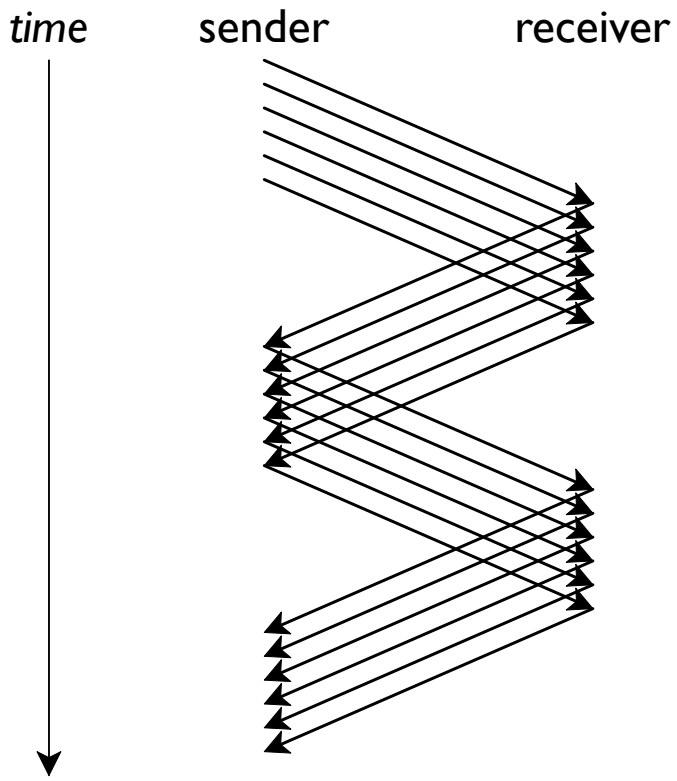


Sequential Messages

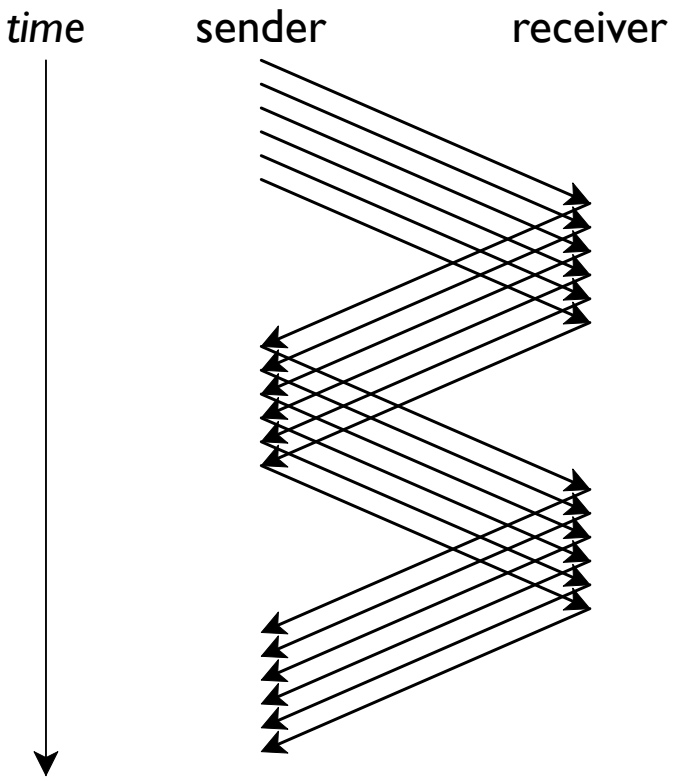


Throughput is limited by latency

Pipelined Messages



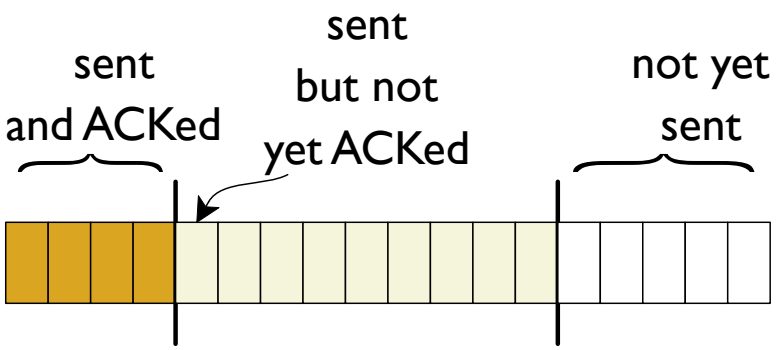
Pipelined Messages



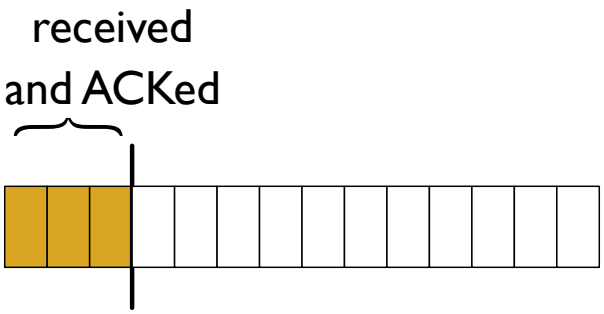
Need a way to track multiple messages in flight

Buffers

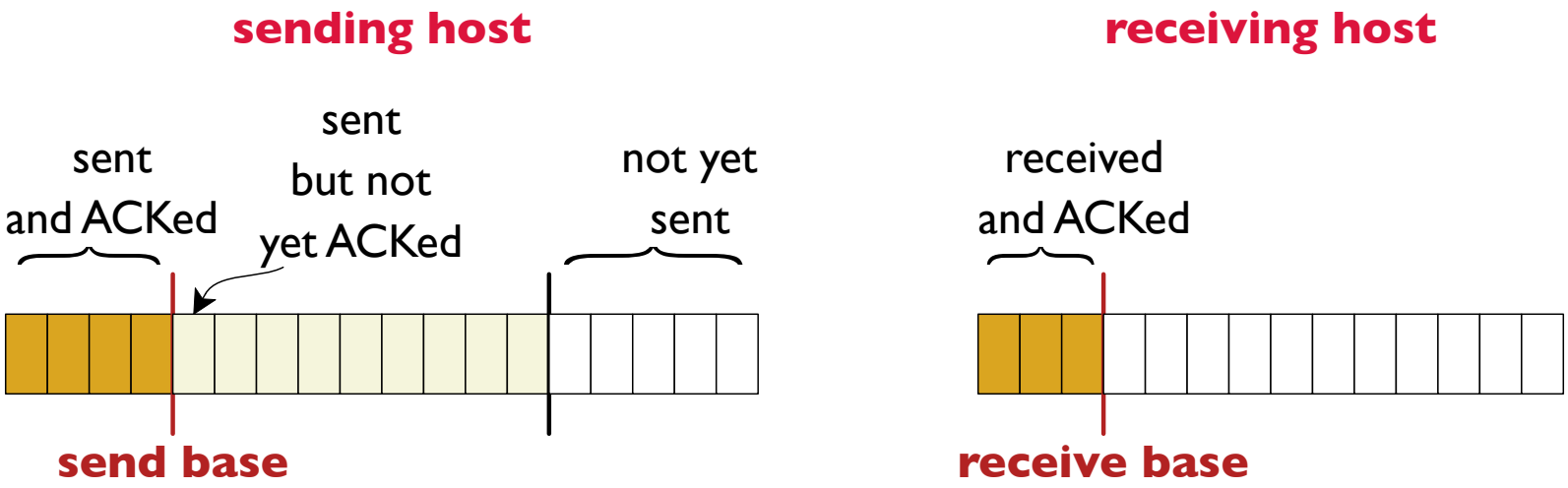
sending host



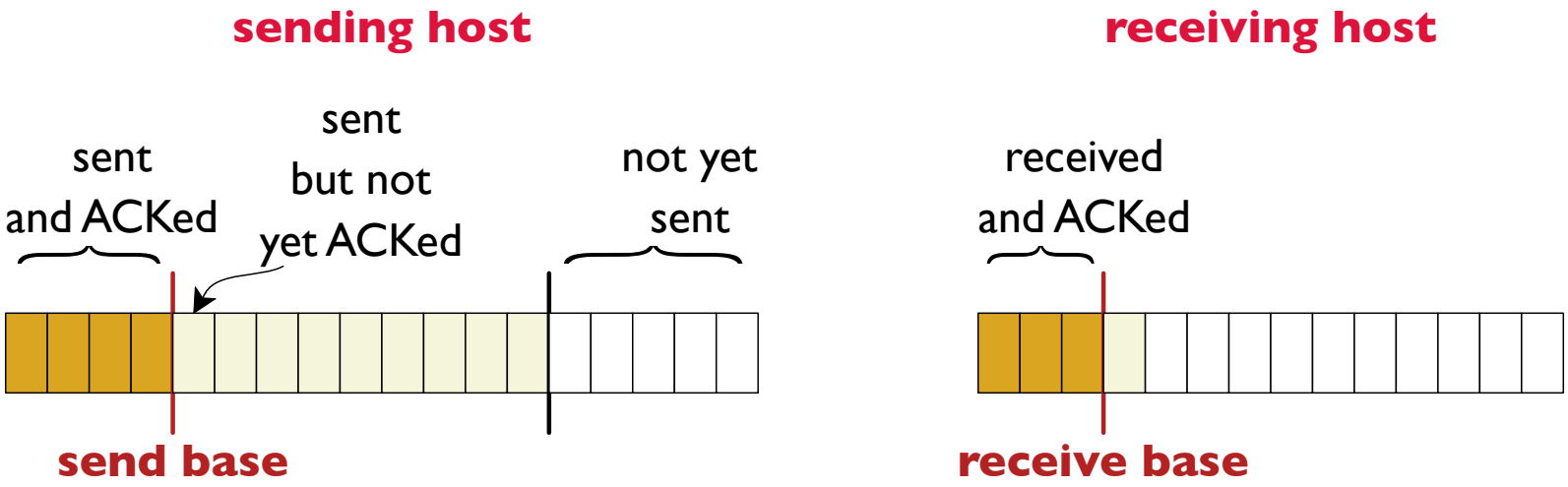
receiving host



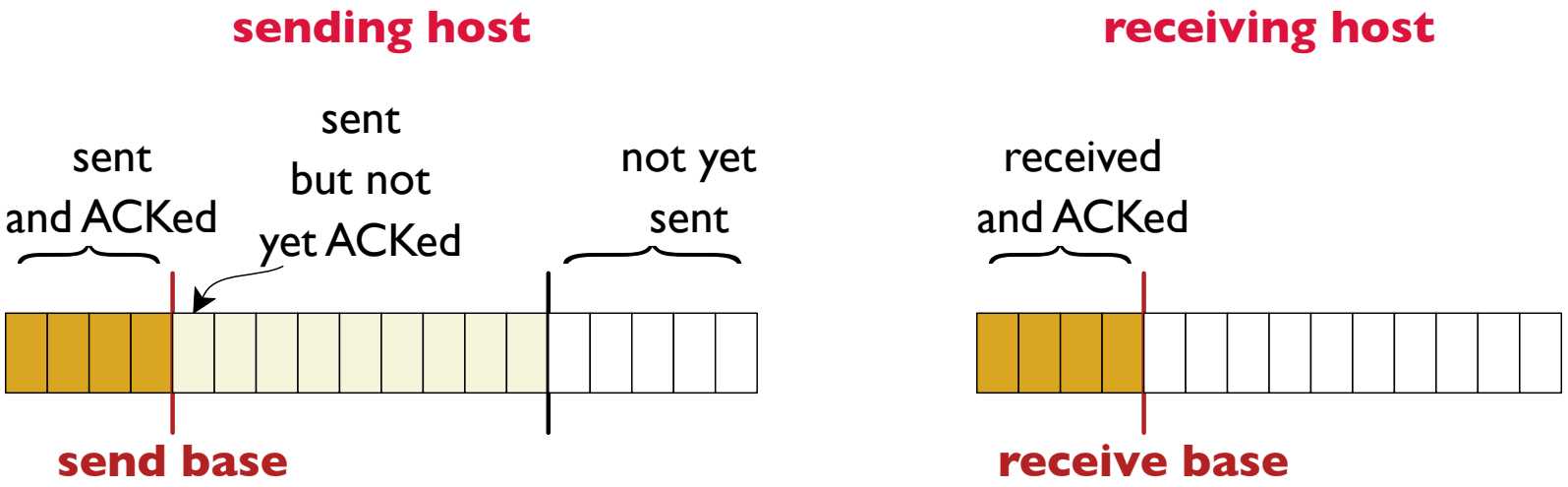
Buffers


sending host
receiving host
sent
and ACKed
sent
but not
yet ACKed
not yet
sent
received
and ACKed
send base
receive base

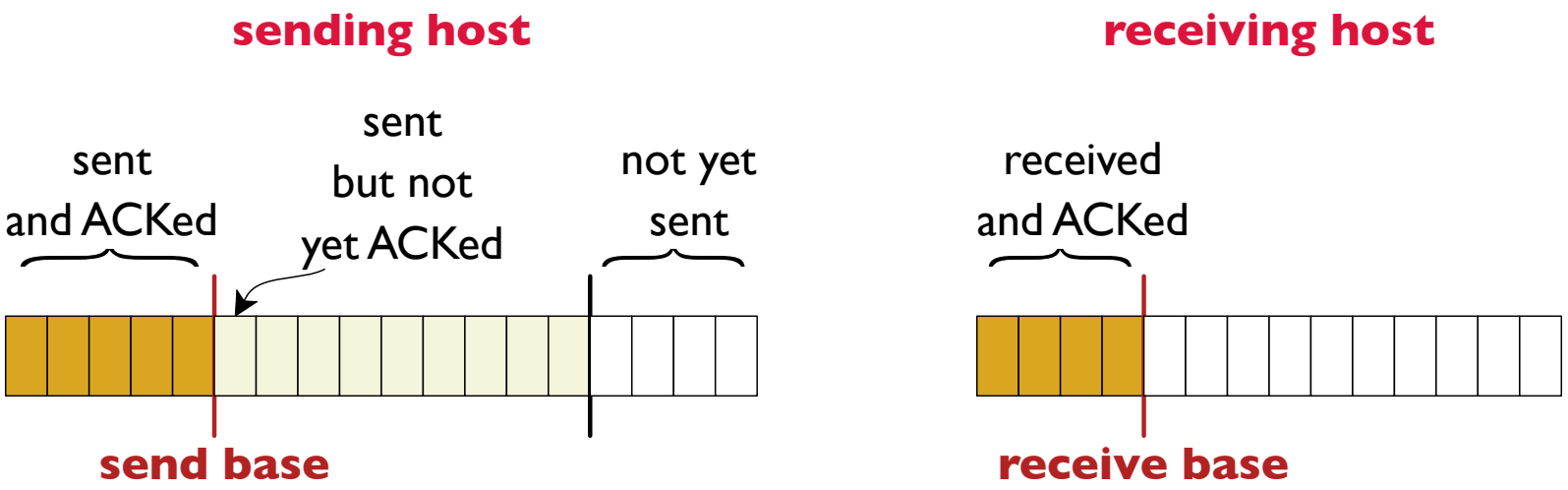
Buffers


sending host
receiving host
sent
and ACKed
sent
but not
yet ACKed
not yet
sent
received
and ACKed
send base
receive base

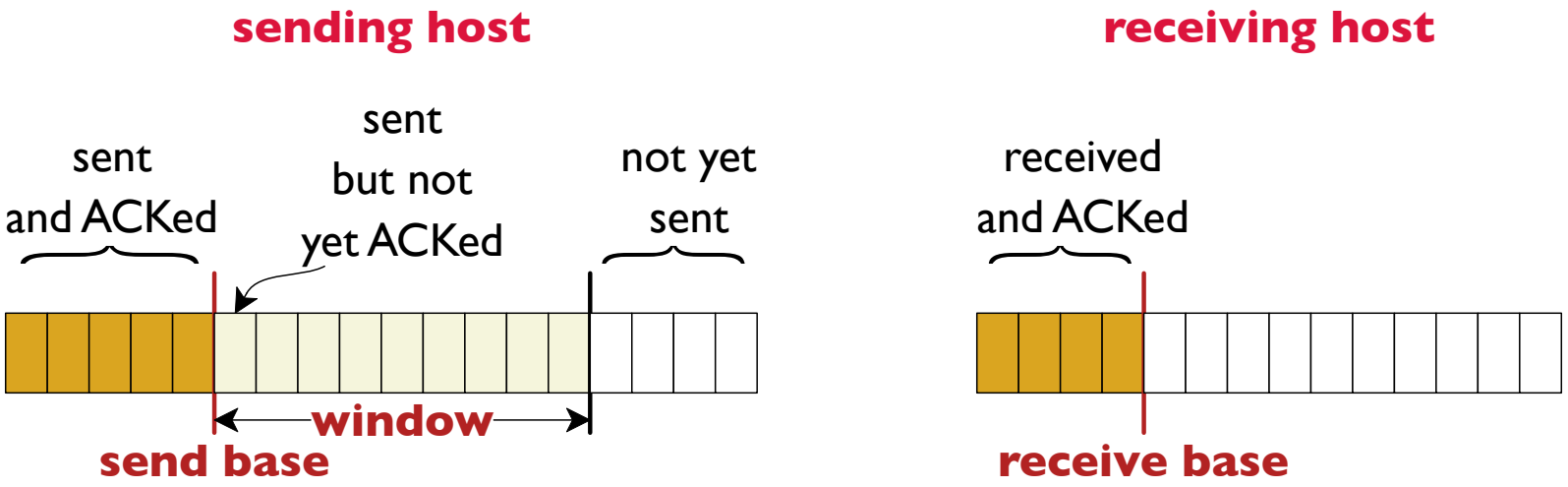
Buffers


sending host
receiving host
sent
and ACKed
sent
but not
yet ACKed
not yet
sent
received
and ACKed
send base
receive base

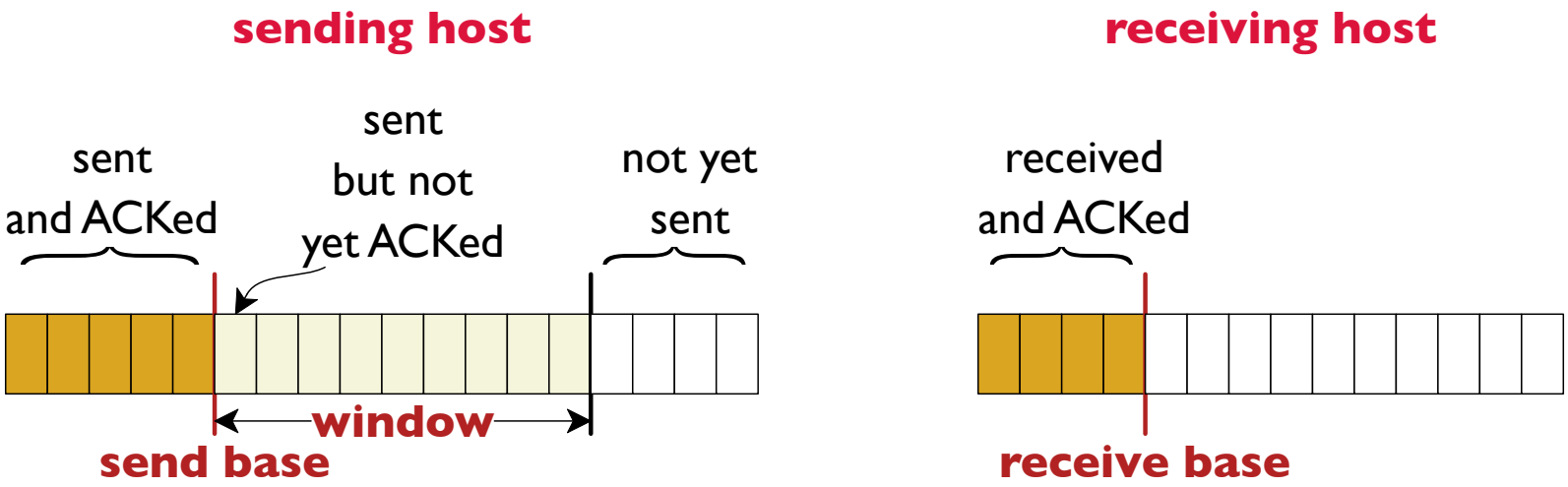
Buffers


sending host
receiving host
sent
and ACKed
sent
but not
yet ACKed
not yet
sent
received
and ACKed
send base
receive base

Buffers

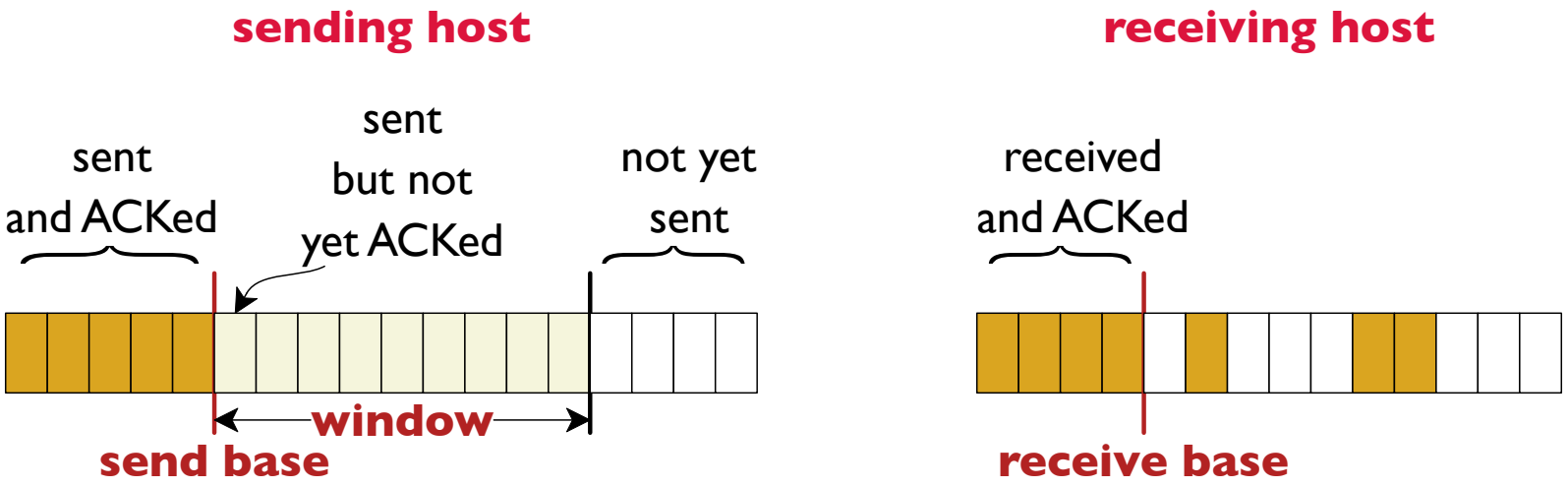

sending host
receiving host
sent
and ACKed
sent
but not
yet ACKed
not yet
sent
received
and ACKed
window
send base
receive base

Buffers

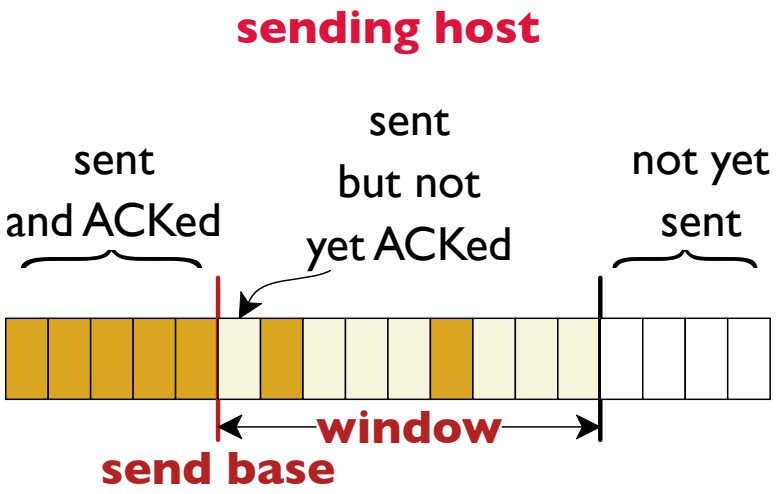


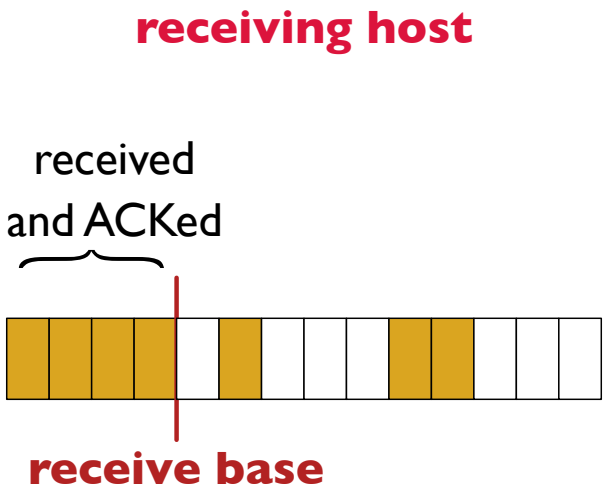
Like a timeout, the window size needs to be chosen well

Buffers

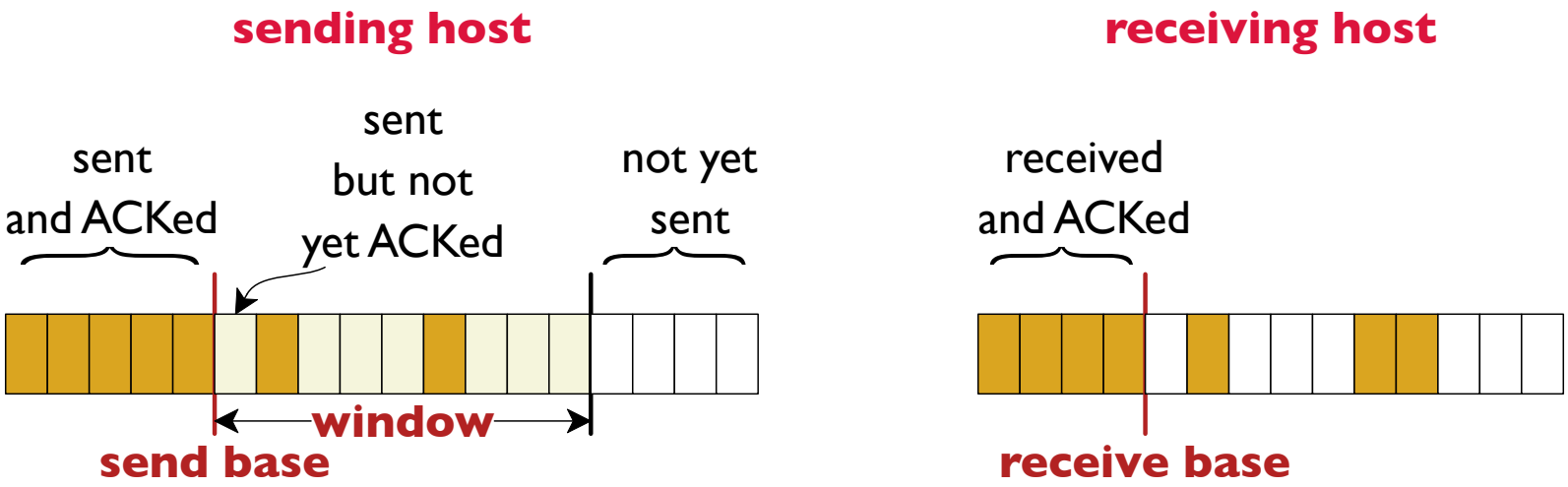

sending host
receiving host
sent
and ACKed
sent
but not
yet ACKed
not yet
sent
received
and ACKed
window
send base
receive base

Buffers


sending host
sent
and ACKed
sent
but not
yet ACKed
not yet
sent
window
send base

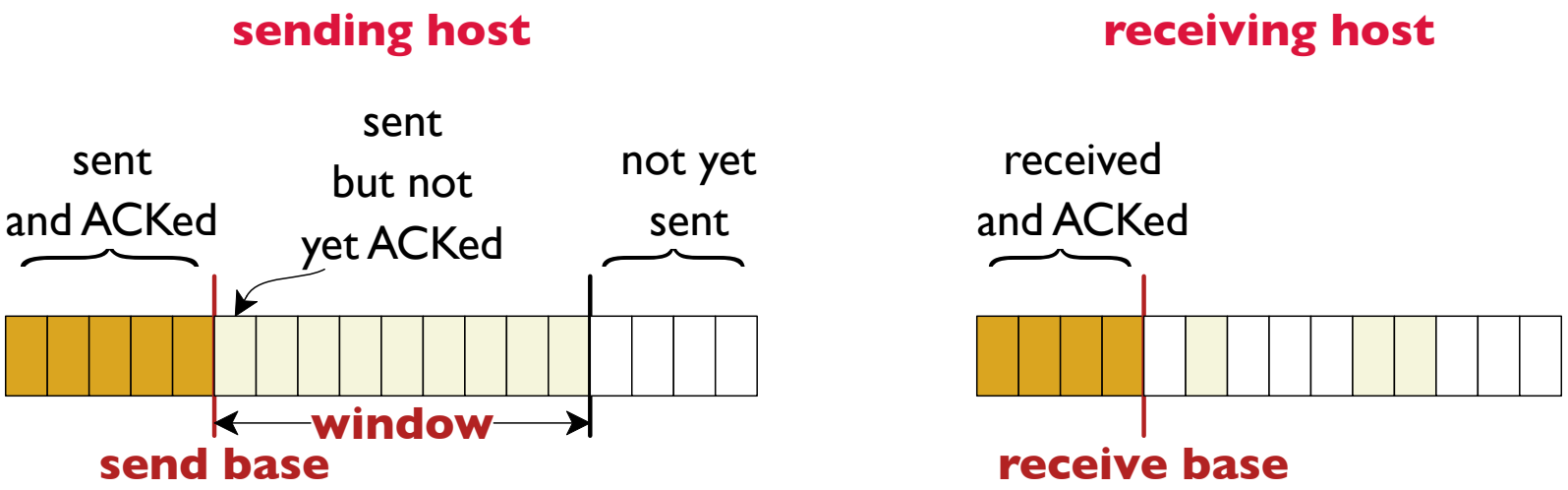

receiving host
received
and ACKed
receive base

Buffers



Go-Back-N : on timeout, re-send in window

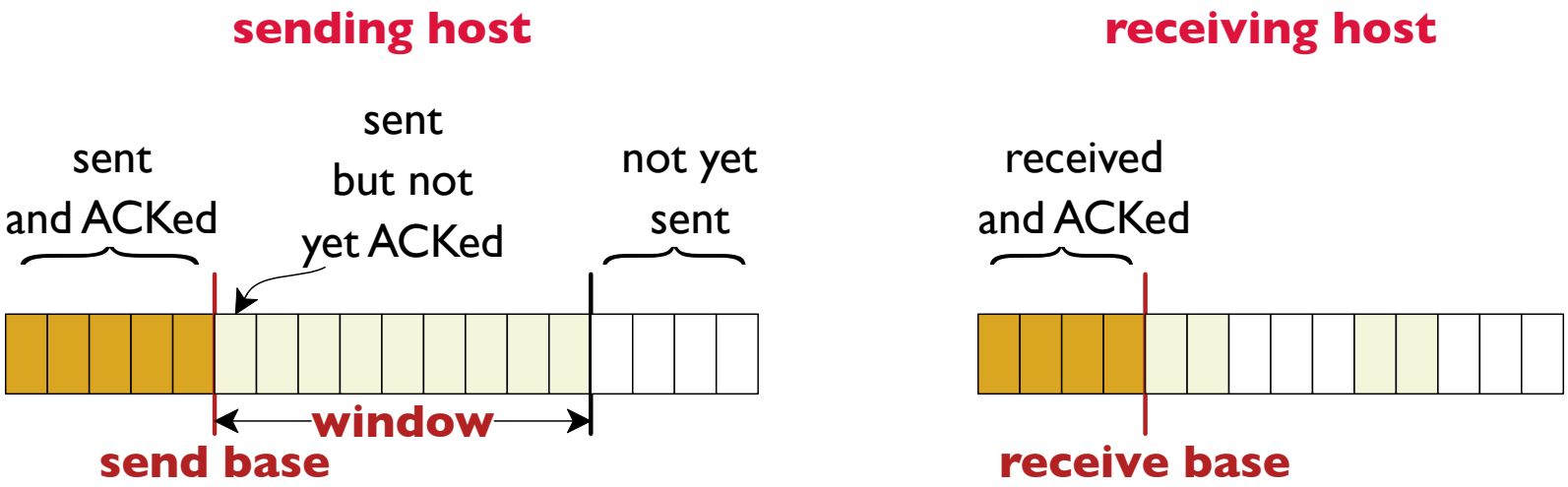
Buffers



Go-Back-N : on timeout, re-send in window

Can use a **cumulative** ACK

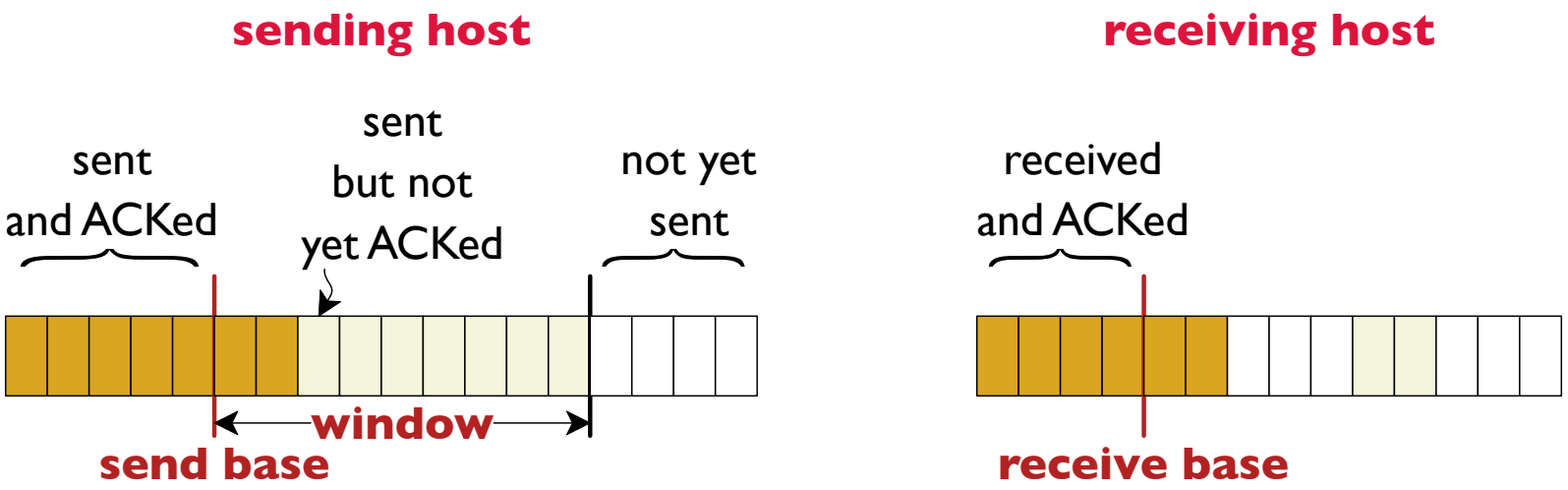
Buffers



Go-Back-N : on timeout, re-send in window

Can use a **cumulative** ACK

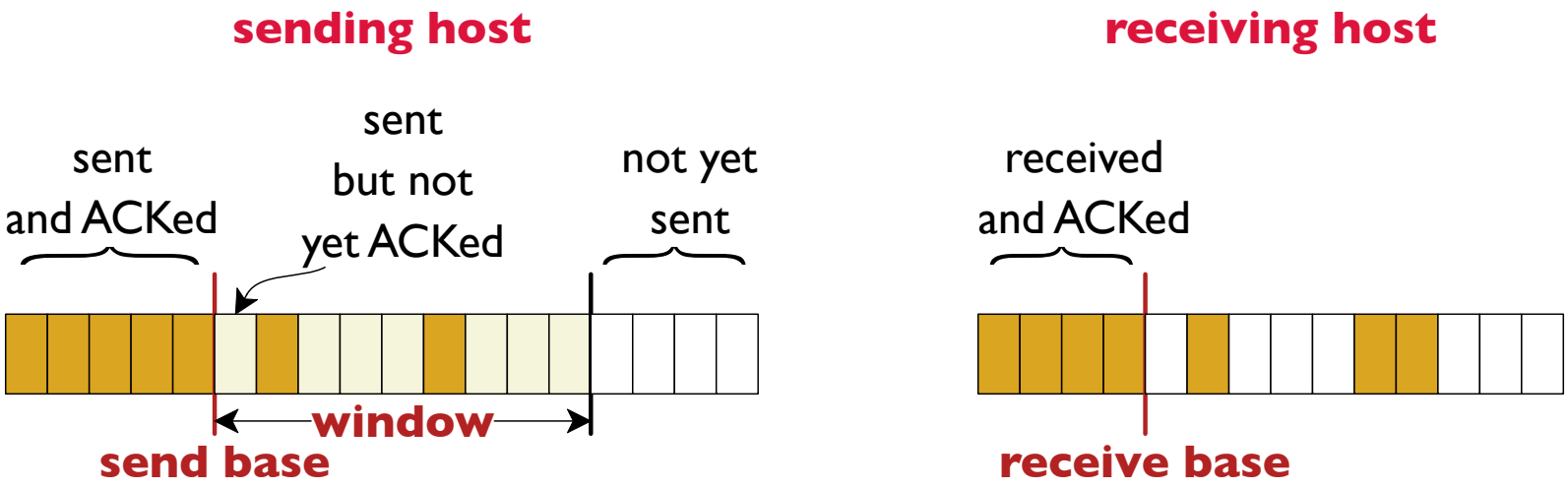
Buffers



Go-Back-N : on timeout, re-send in window

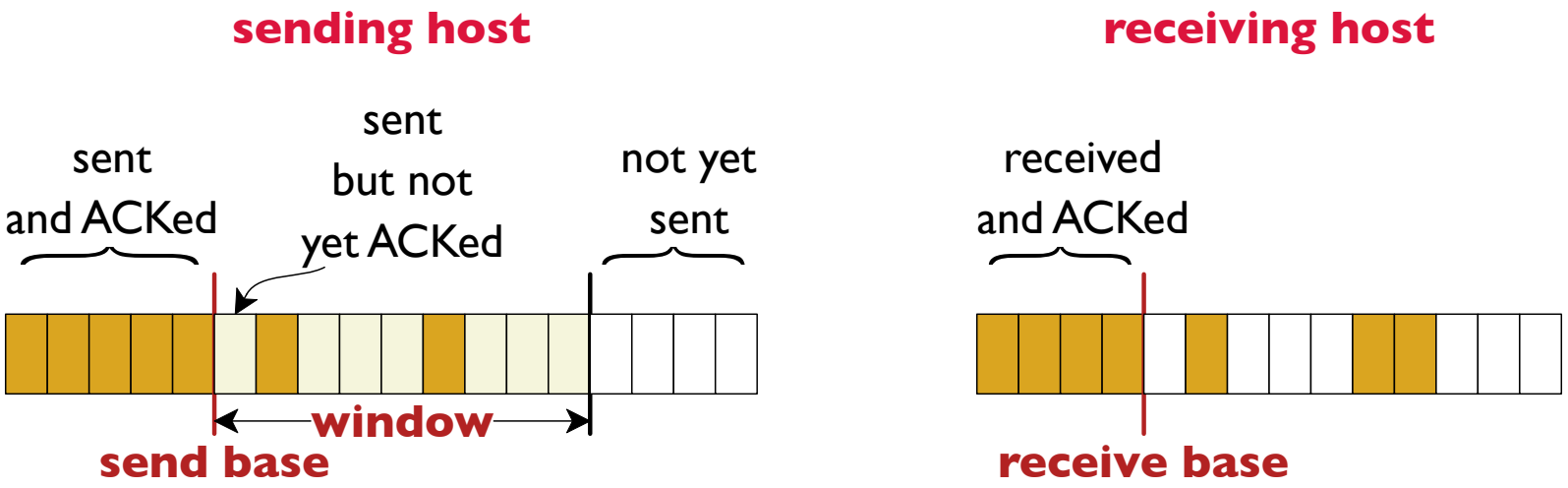
Can use a **cumulative** ACK

Buffers



Selective repeat : on timeout, re-send unACKed

Buffers



Selective repeat : on timeout, re-send unACKed

Each packet must be specifically ACKed