Distributed Systems – Efficient Communication Software

Today

✓ Distributed System Basics

✓ Lamport's clock

⇒ Efficient Communication software

Therkath + Levy
- * Application interface to the kernel
- * Inside the kernel

Wetherall
- * End-to-End QoS via Active Networks

Friday

Synthesizing Network Protocol Stacks

Latency Vs. Throughput

Latency
- Elapsed time

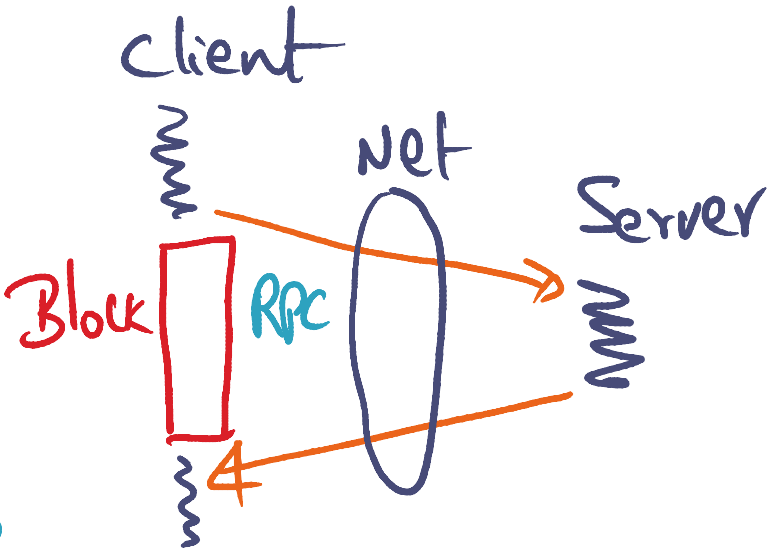
Throughput
- Events per unit time
- Bandwidth: thruput measure

Latency Vs. Throughput

Latency
- Elapsed time

Throughput
- Events per unit time
- Bandwidth: thruput measure

RPC performance

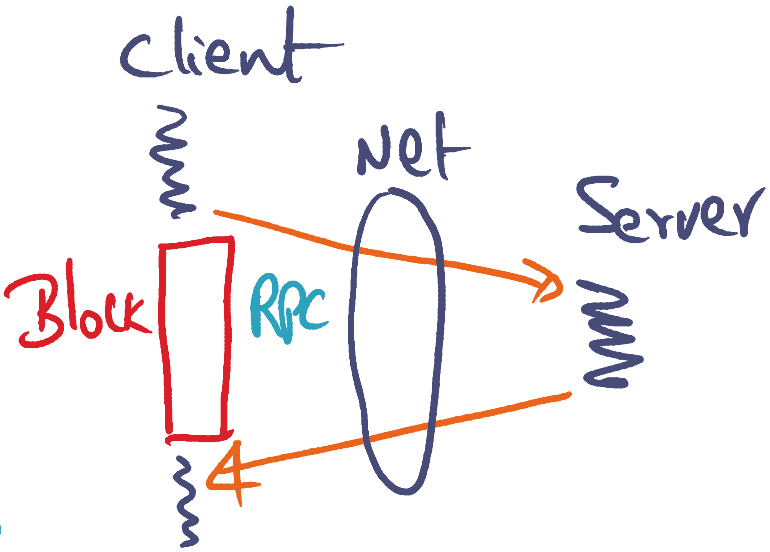


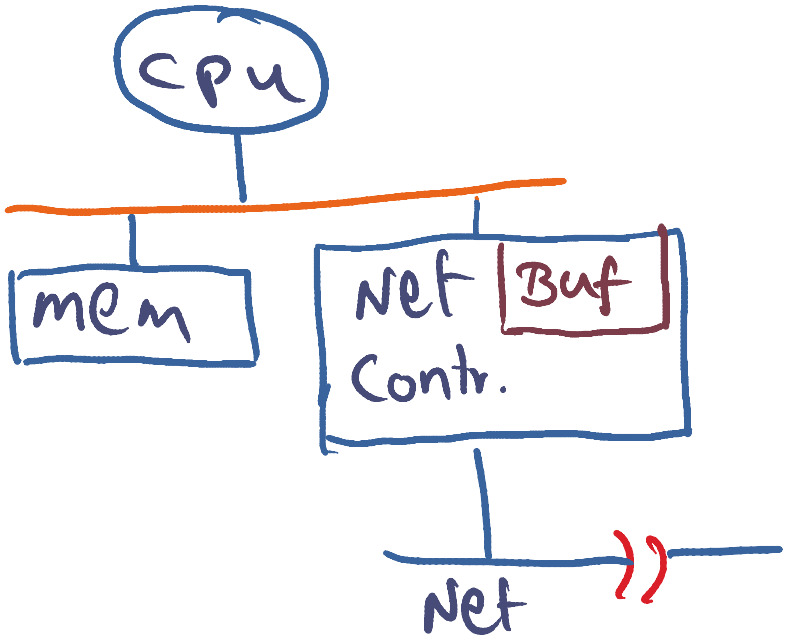
Latency Vs. Throughput

Latency
- Elapsed time

Throughput
- Events per unit time
- Bandwidth: thruput measure

RPC performance
- Hardware overhead
- Software overhead





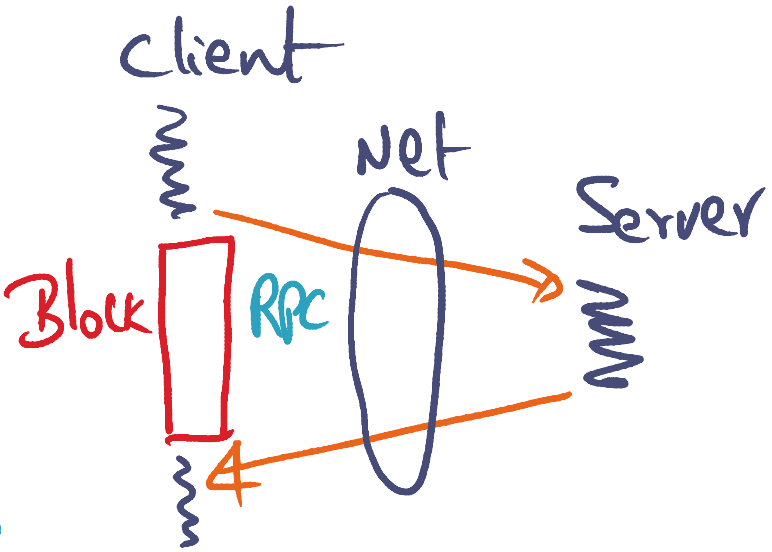
Latency Vs. Throughput

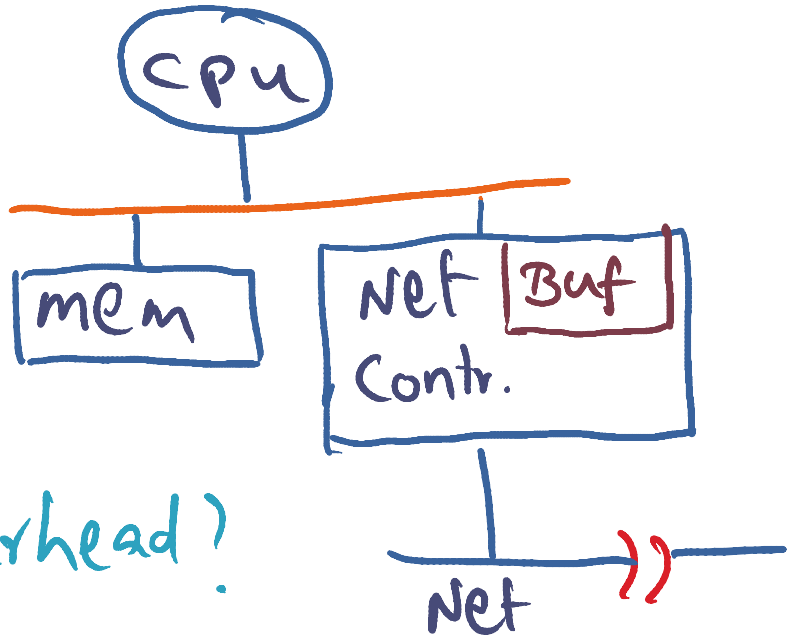
Latency
- Elapsed time

Throughput
- Events per unit time
- Bandwidth: thruput measure

RPC performance
- Hardware overhead
- software overhead

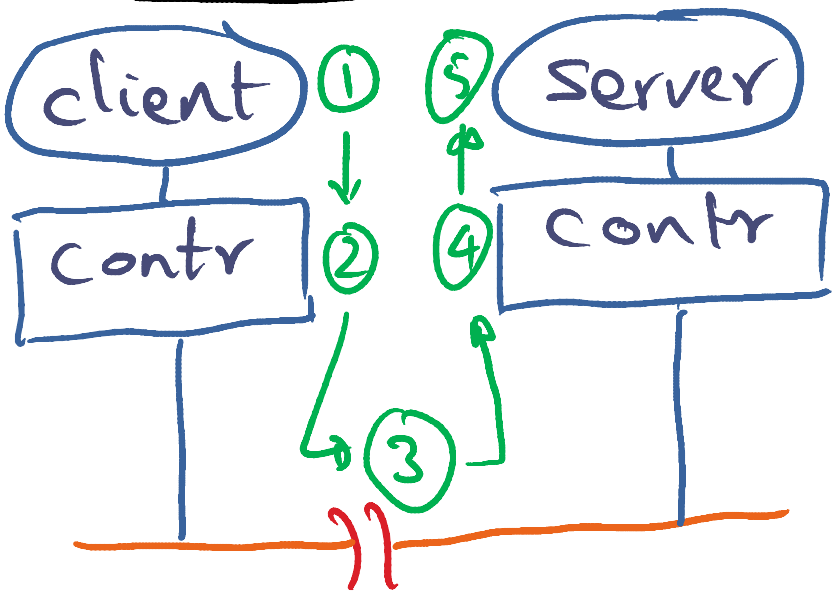
Focus
- How to reduce software overhead?



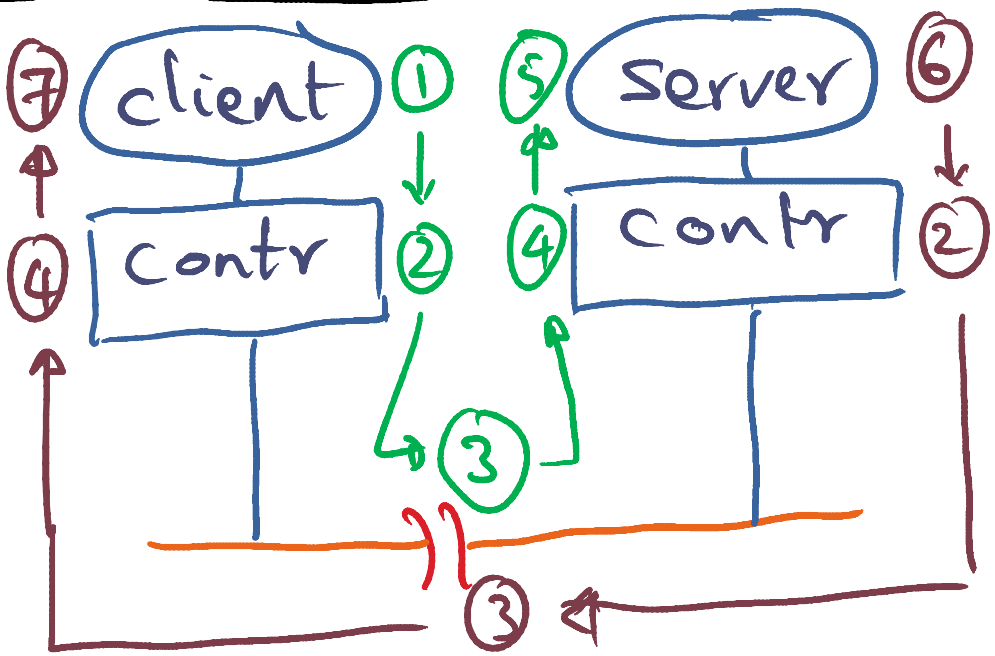


Components of RPC Latency

1. client call
2. Controller latency
3. Time on wire
4. Interrupt handling
5. server setup to execute call



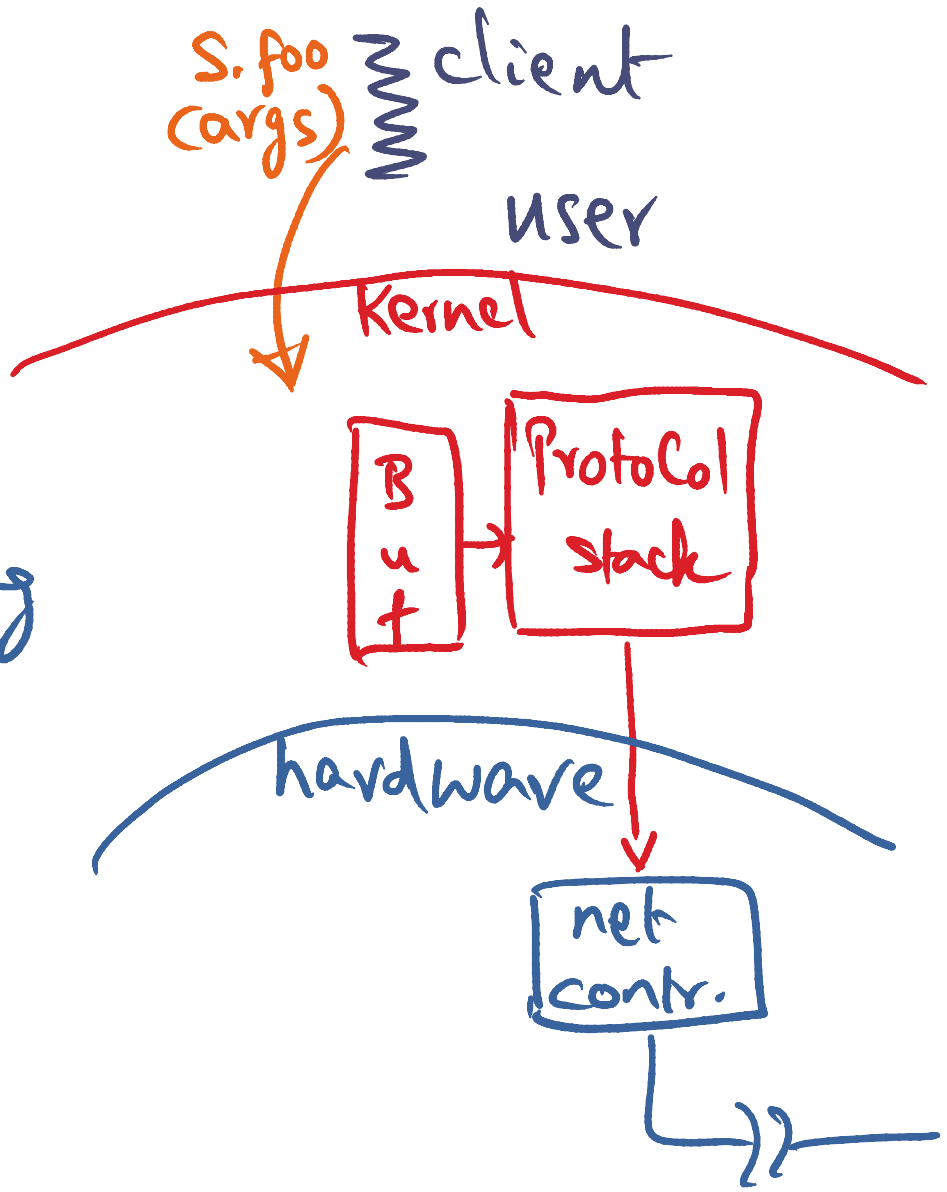
Components of RPC Latency



1. client call
2. Controller latency
3. Time on Wire
4. Interrupt handling
5. server setup to execute call
6. server execution + reply
7. client setup to receive results & restart

Sources of overhead in RPC

- marshaling
- Data copying
- Control transfer
- Protocol processing



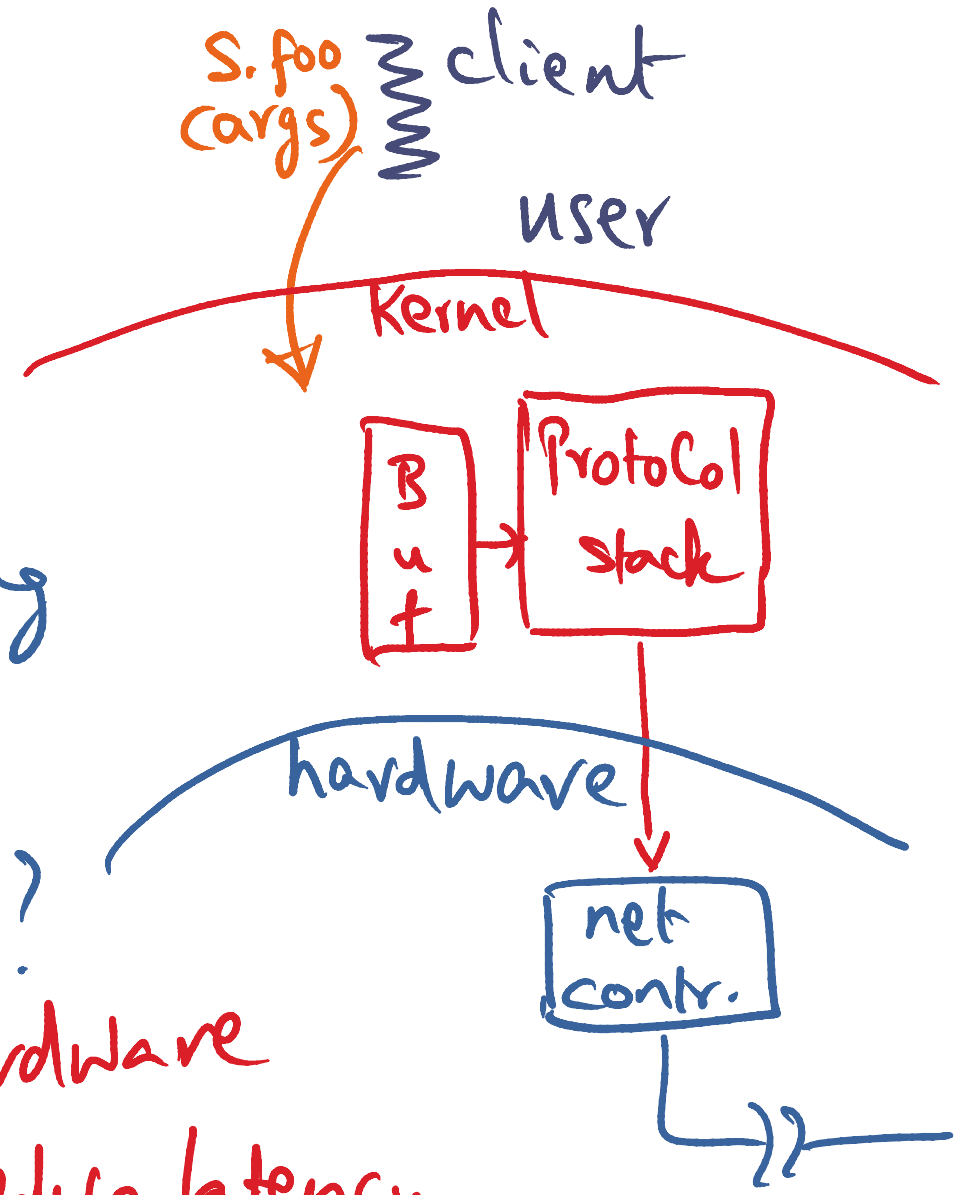
Sources of overhead in RPC

marshaling

Data copying

Control transfer

Protocol processing

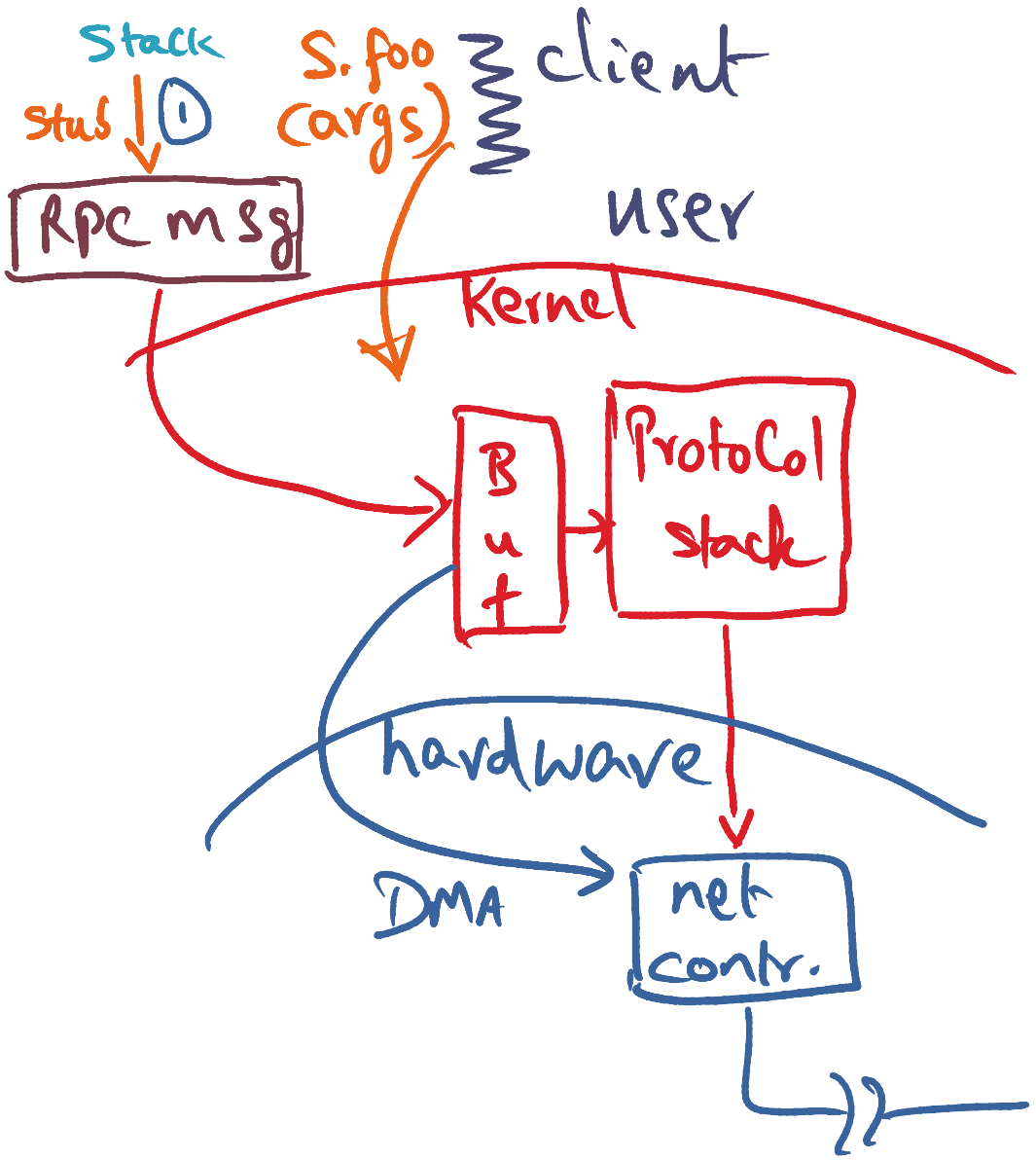


How to reduce Kernel overhead?

- Take what hardware gives you to reduce latency

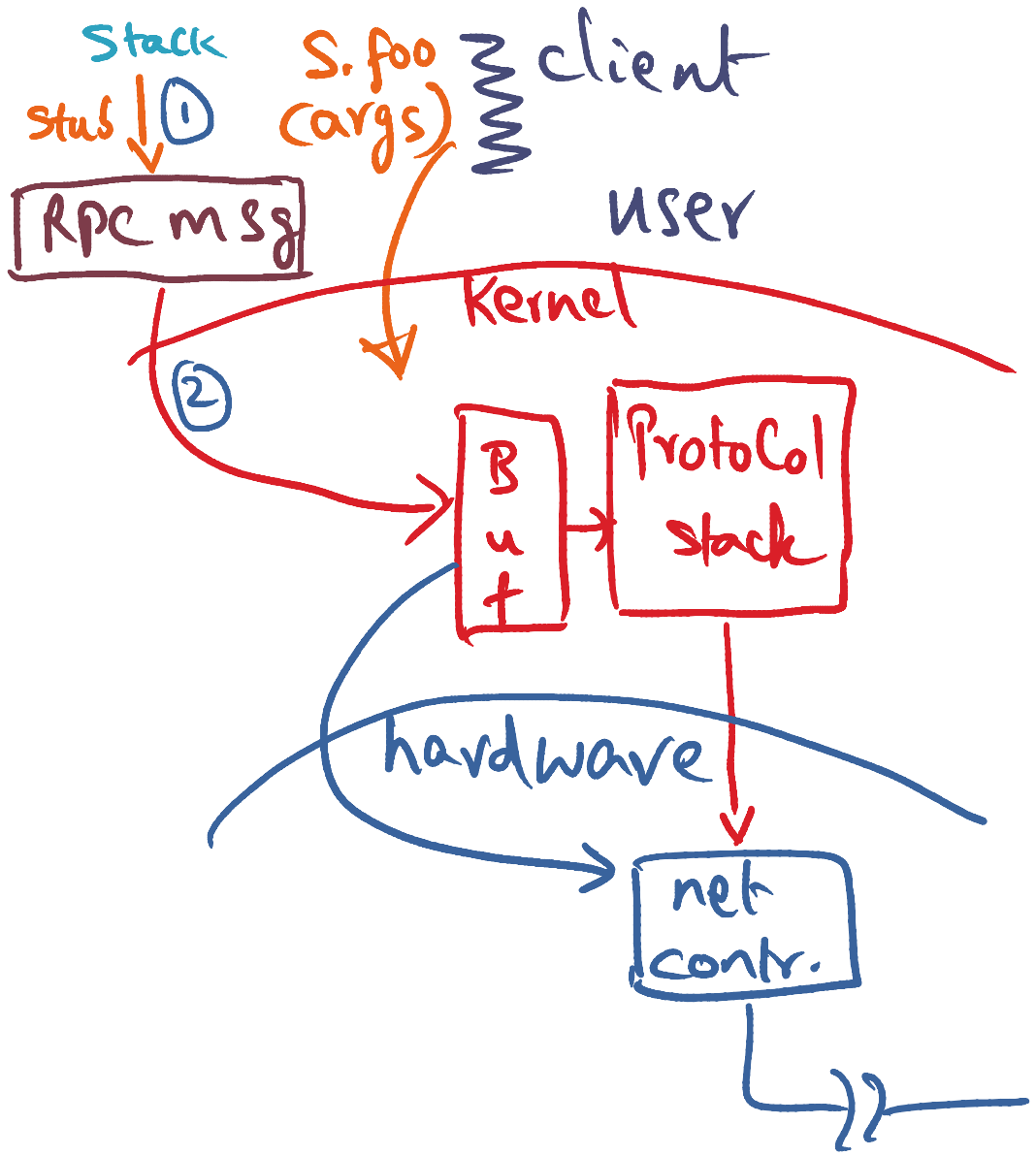
marshaling and data copying

Three copies
- client stub



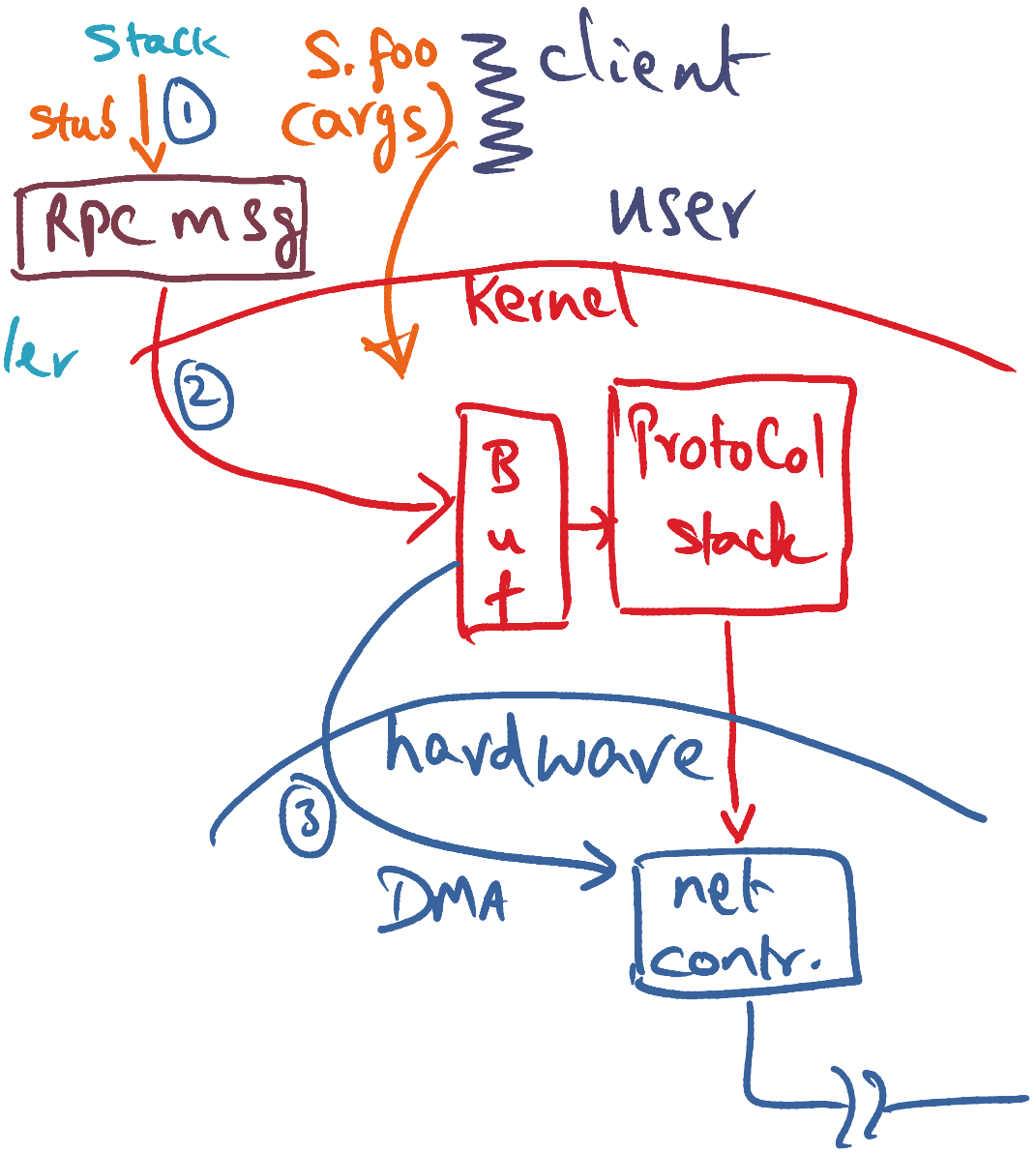
marshaling and data copying

Three copies
- client stub
- kernel buffer



marshaling and data copying

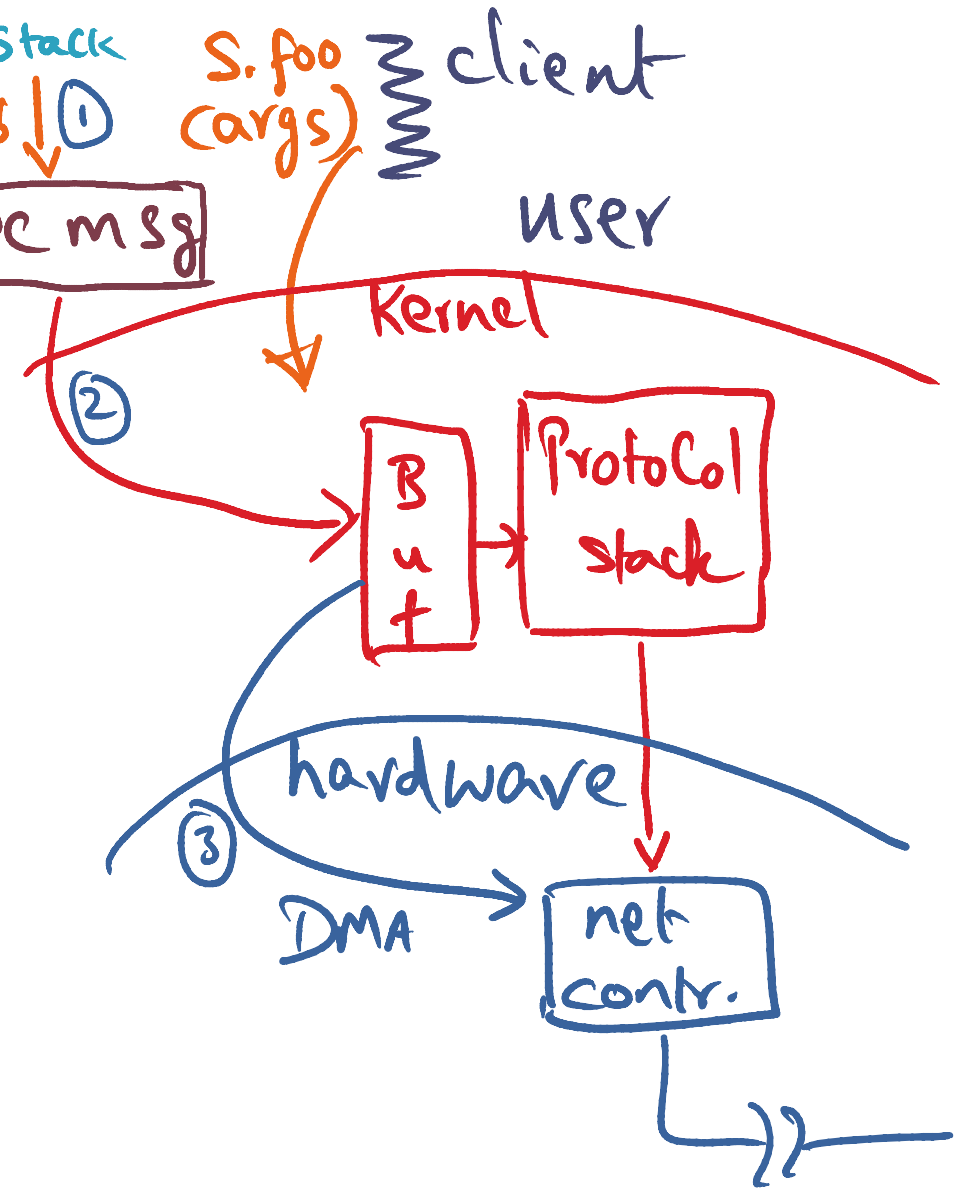
Three copies
- client stub
- kernel buffer
- DMA to controller



marshaling and data copying

Three copies
- client stub
- kernel buffer
- DMA to controller

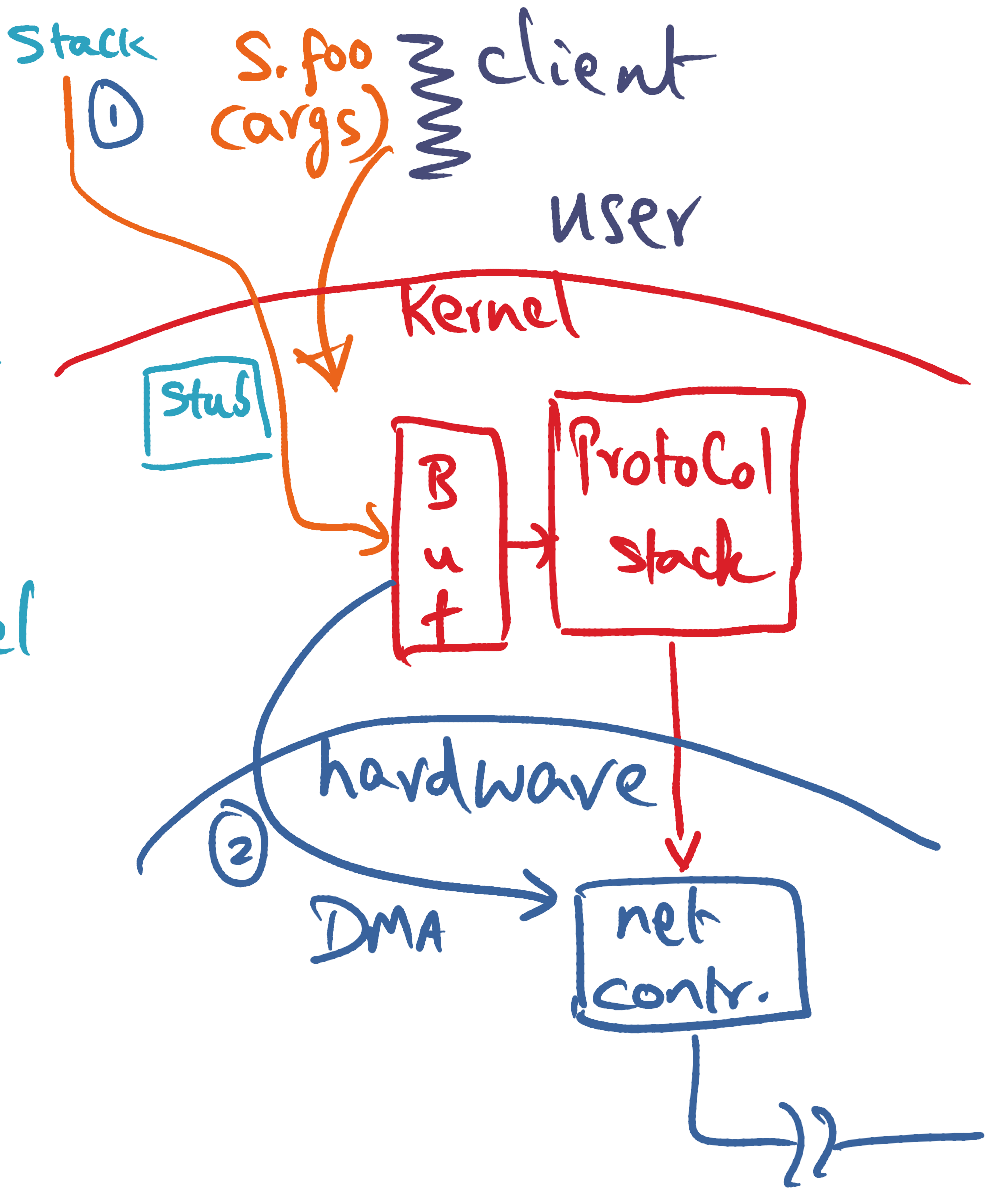
Reducing copies?



marshaling and data copying

Three copies
- client stub
- kernel buffer
- DMA to controller

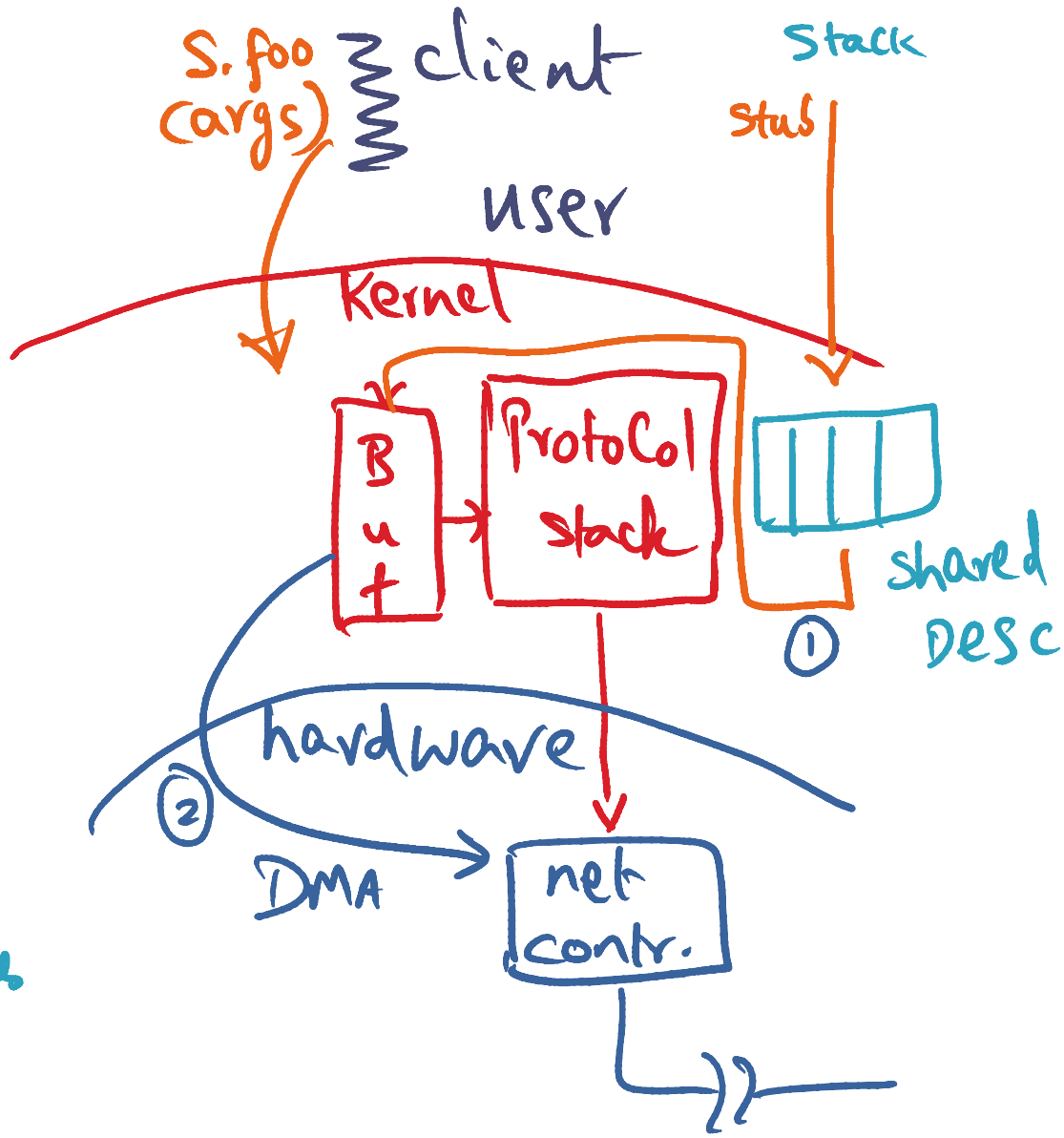
Reducing copies?
- marshal into kernel buffer directly

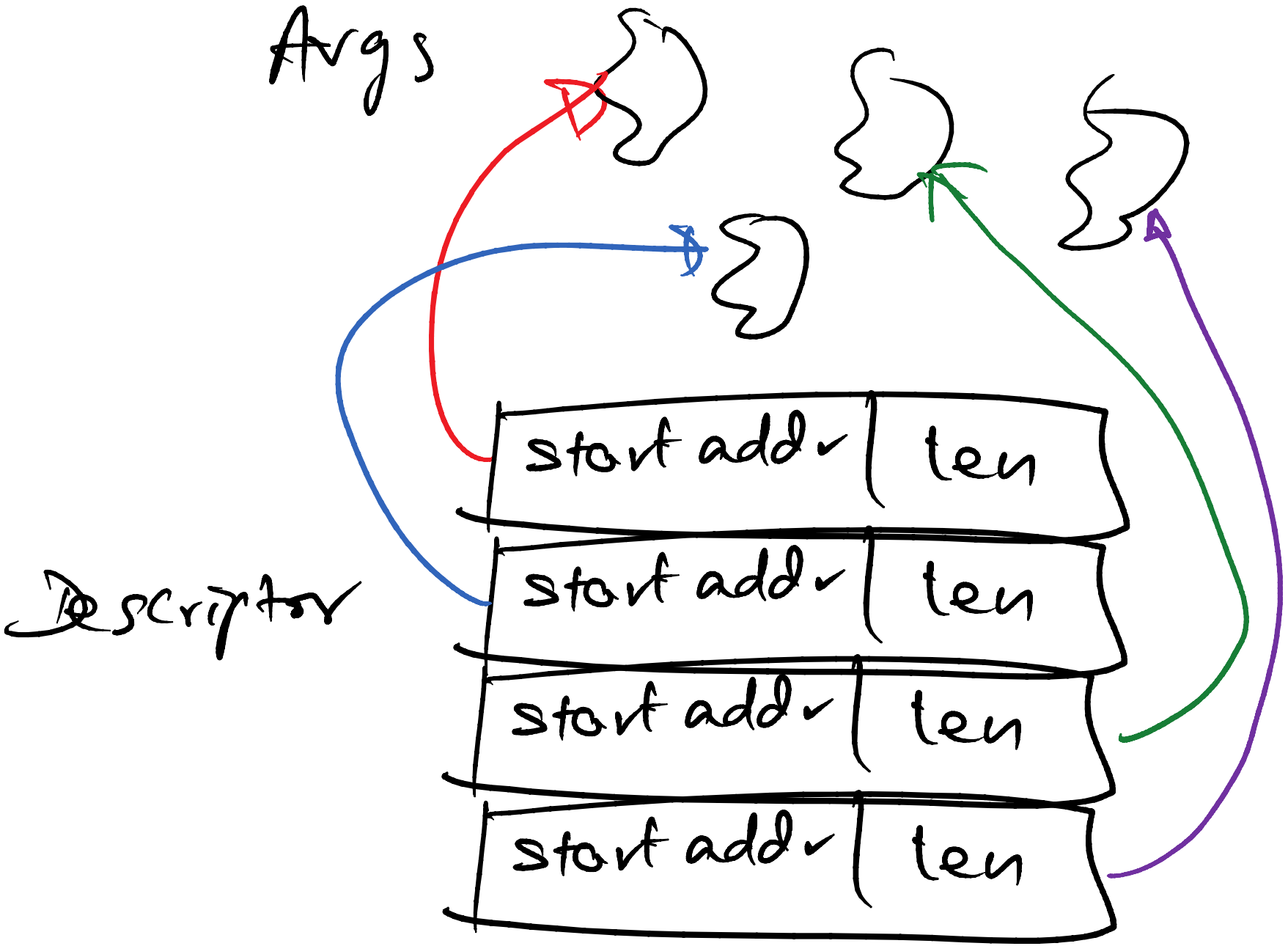


marshaling and data copying

Three copies
- client stub
- kernel buffer
- DMA to controller

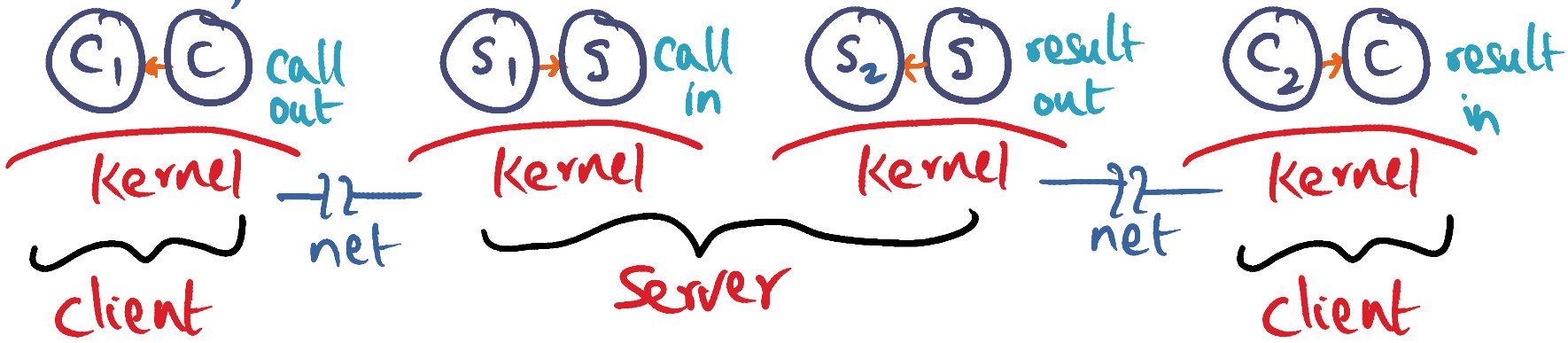
Reducing copies?
- marshal into kernel buffer directly
  OR
- shared descriptors between client stub and kernel




Args
Descriptor
start addr
len
start addr
len
start addr
len
start addr
len

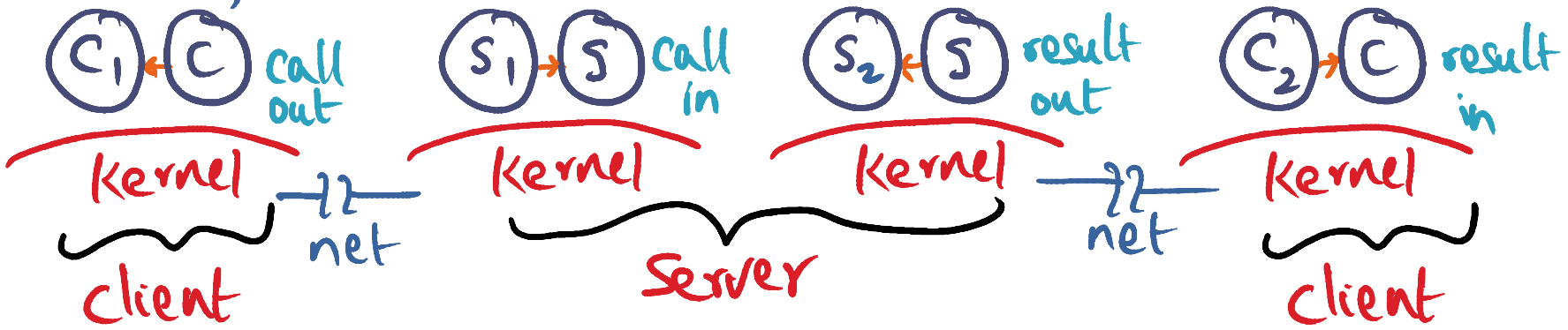
Control Transfer

Potentially 4

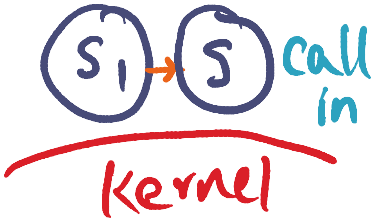


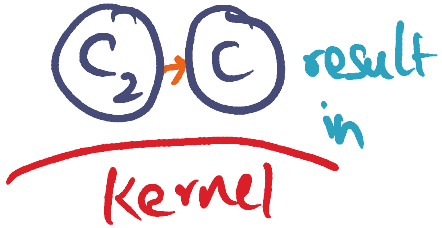
Control Transfer

Potentially 4



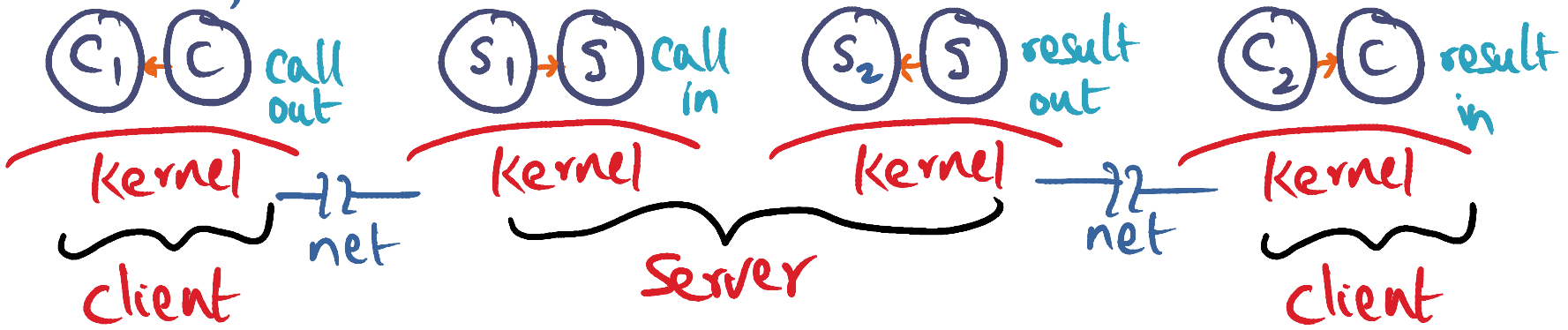
only two in critical path of latency



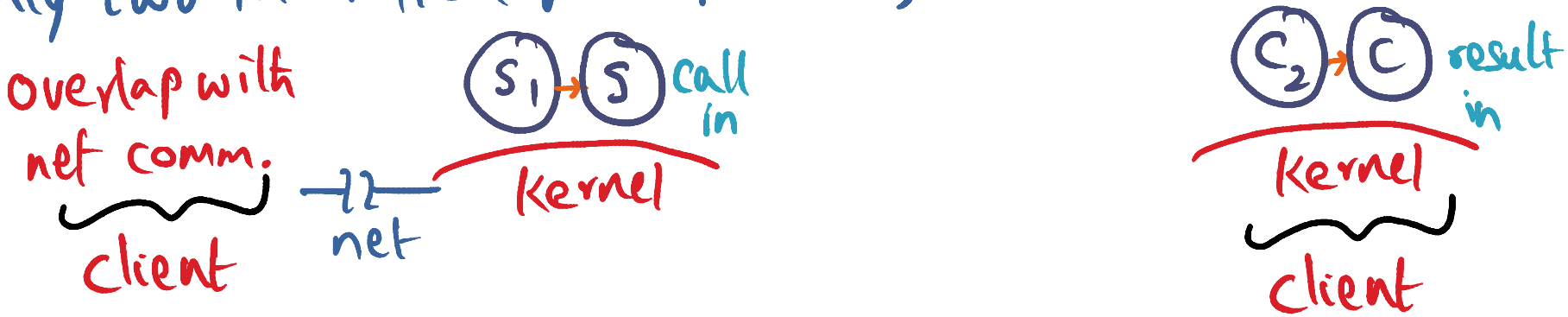


Control Transfer

Potentially 4

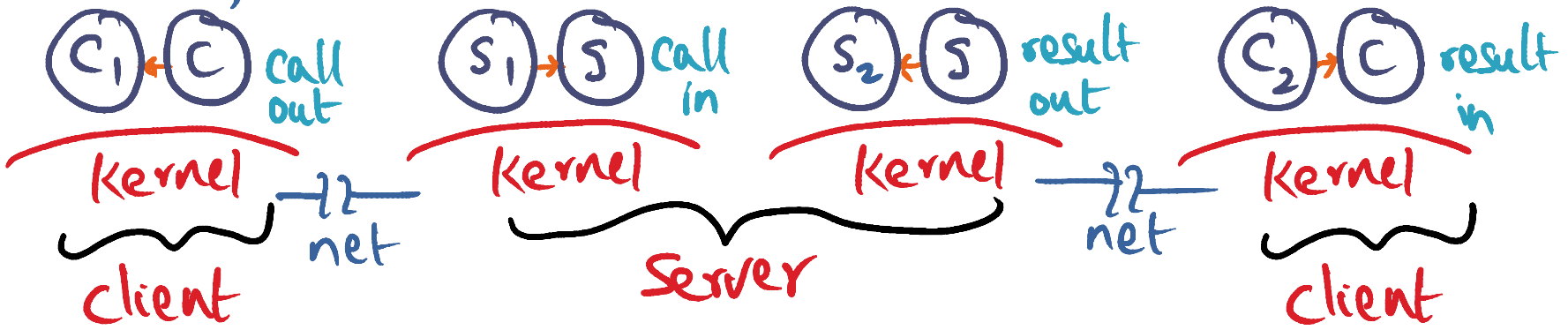


only two in critical path of latency

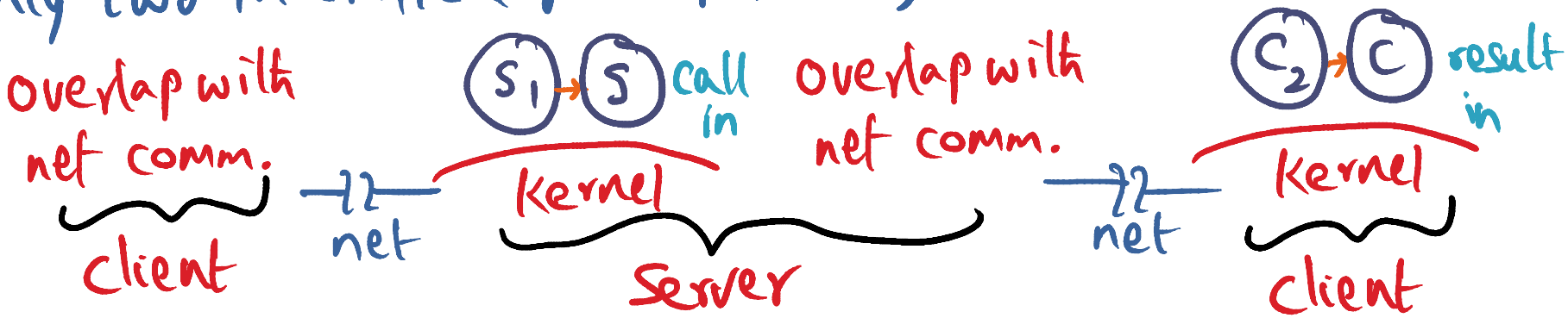


Control Transfer

Potentially 4

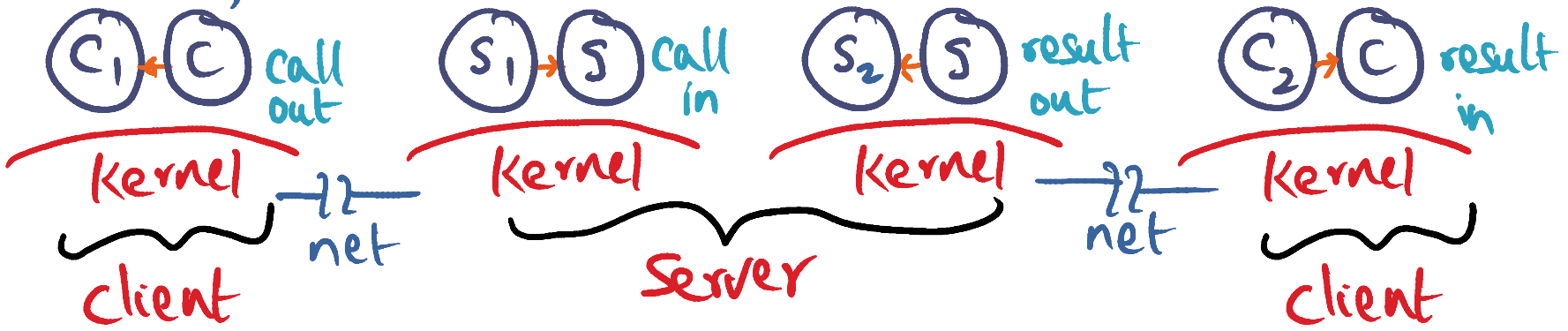


only two in critical path of latency

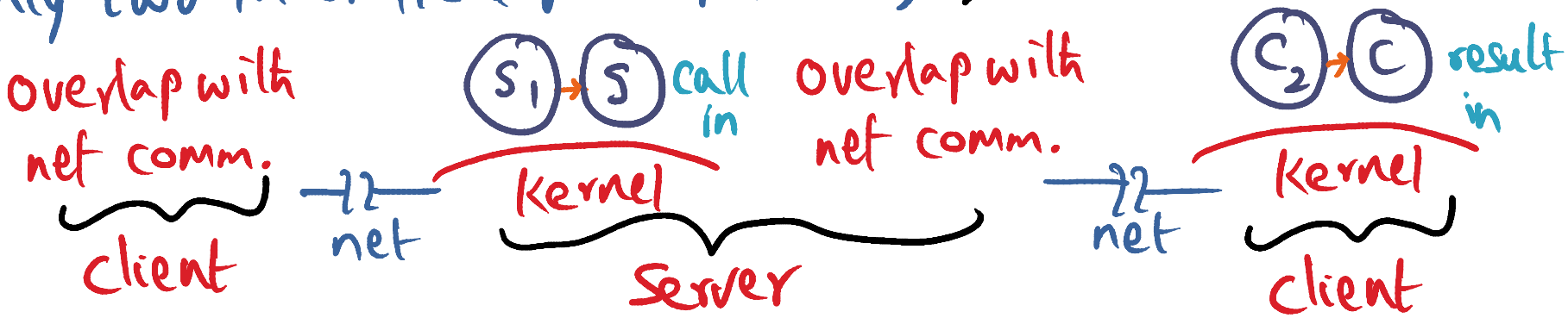


Control Transfer

Potentially 4

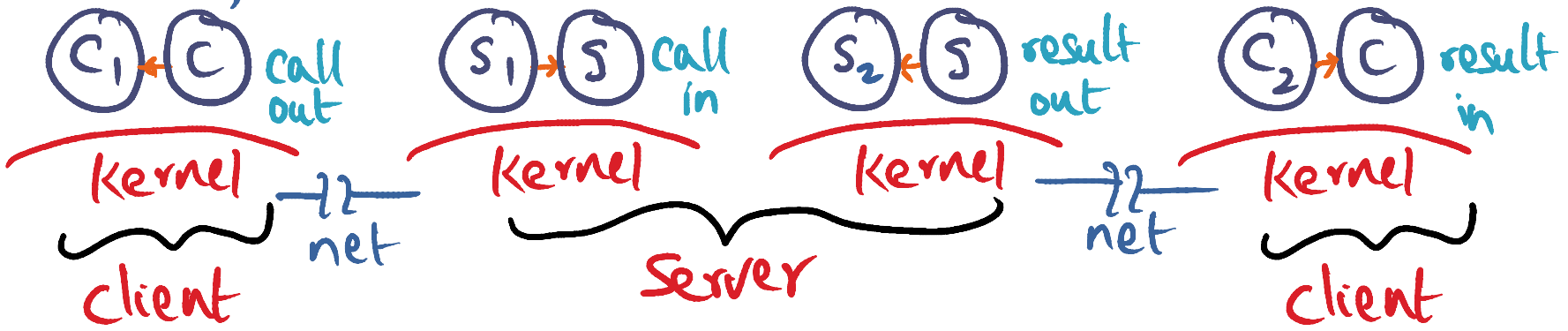


only two in critical path of latency ⇒ down to 2

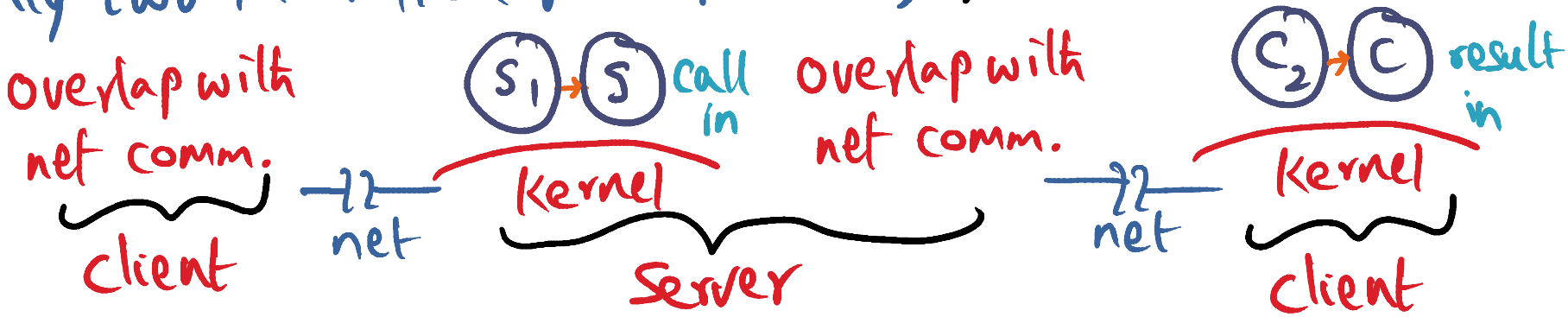


Control Transfer

Potentially 4



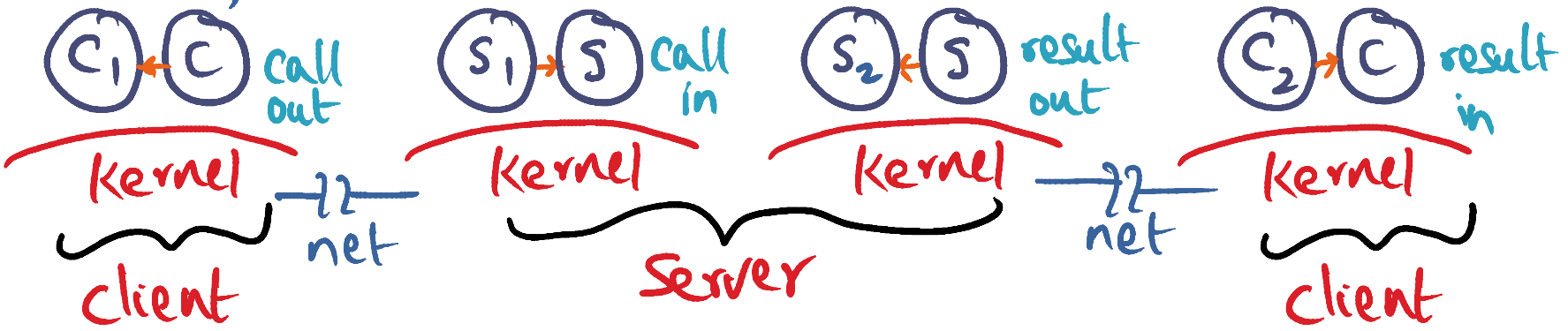
only two in critical path of latency ⇒ down to 2



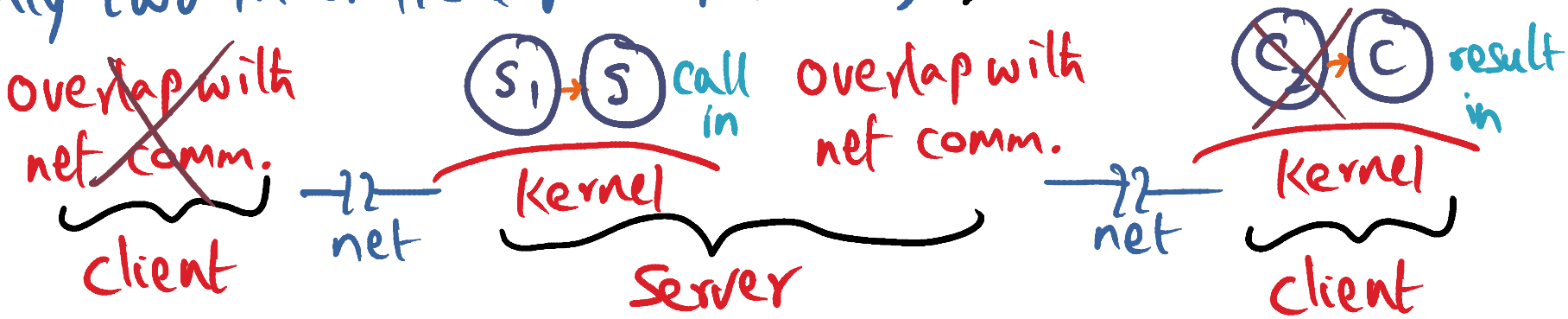
Can we reduce it to 1?

Control Transfer

Potentially 4



only two in critical path of latency ⇒ down to 2



Can we reduce it to 1?

- Yes if we spin instead of switch on client side

Protocol Processing

What transport for RPC?

LAN reliable ⇒ reduce latency

Protocol Processing

What transport for RPC?

- LAN reliable ⇒ reduce latency

choices for reducing latency in transport

- No low level ACKs
- Hardware checksum for packet integrity
- No client side buffering since client blocked
- overlap server side buffering with result transmission

Key Takeaways from Thekkath and Levy Paper

- Latency more important than bandwidth.
- High bandwidth networks do not imply low latency communication.
- It is possible to shave off software overheads and get close to hardware speeds by careful OS design of the RPC code path.

Read the paper by Thekkath and Levy fully to appreciate how they put these principles to practice and build a high-performance RPC system.

Today

✓ Distributed System Basics

✓ Lamport's clock

Efficient Communication software

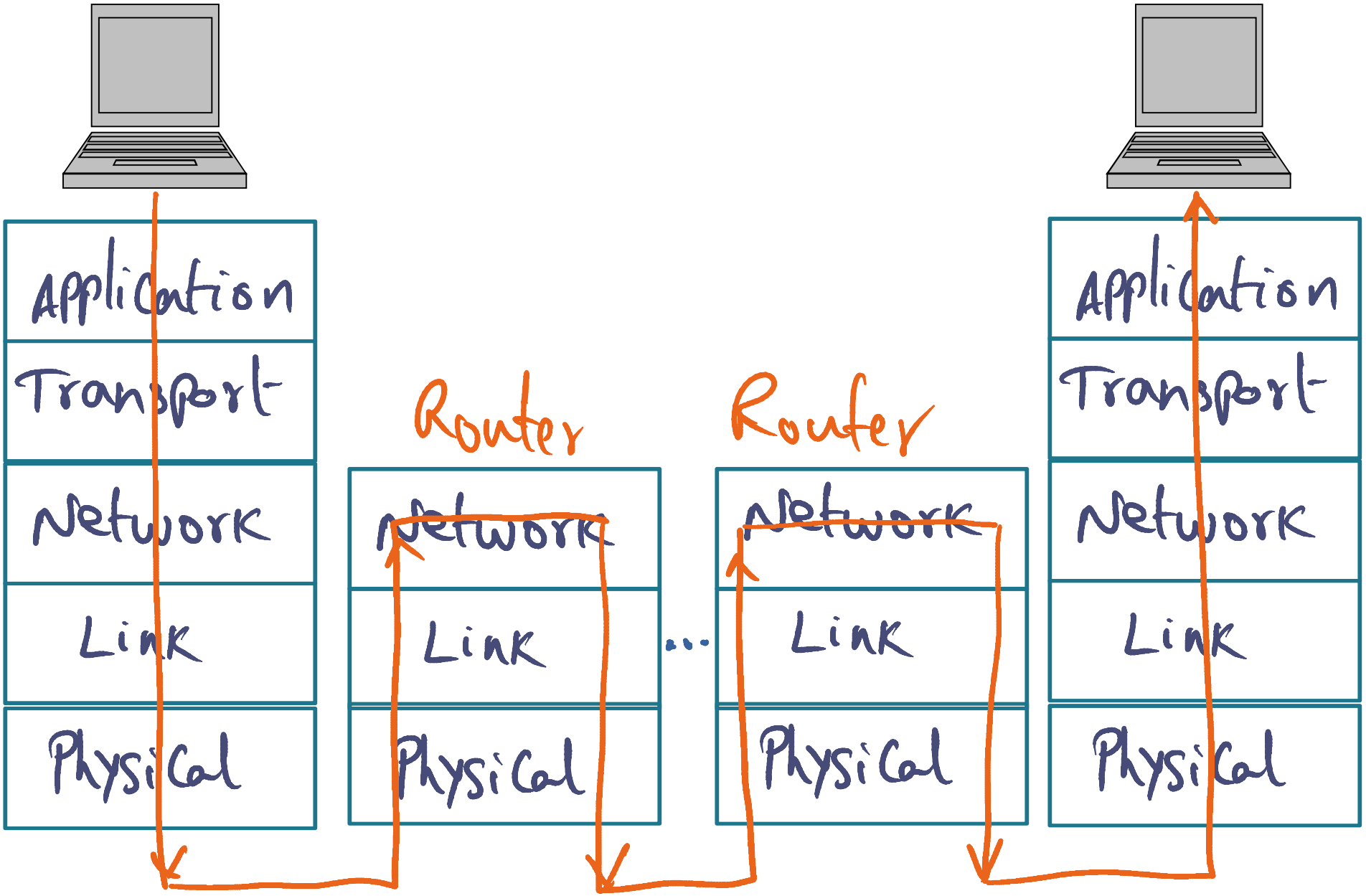
Therkath + Levy
- * Application interface to the Kernel
- * Inside the Kernel

Wetherall
- * End-to-End QoS via Active Networks

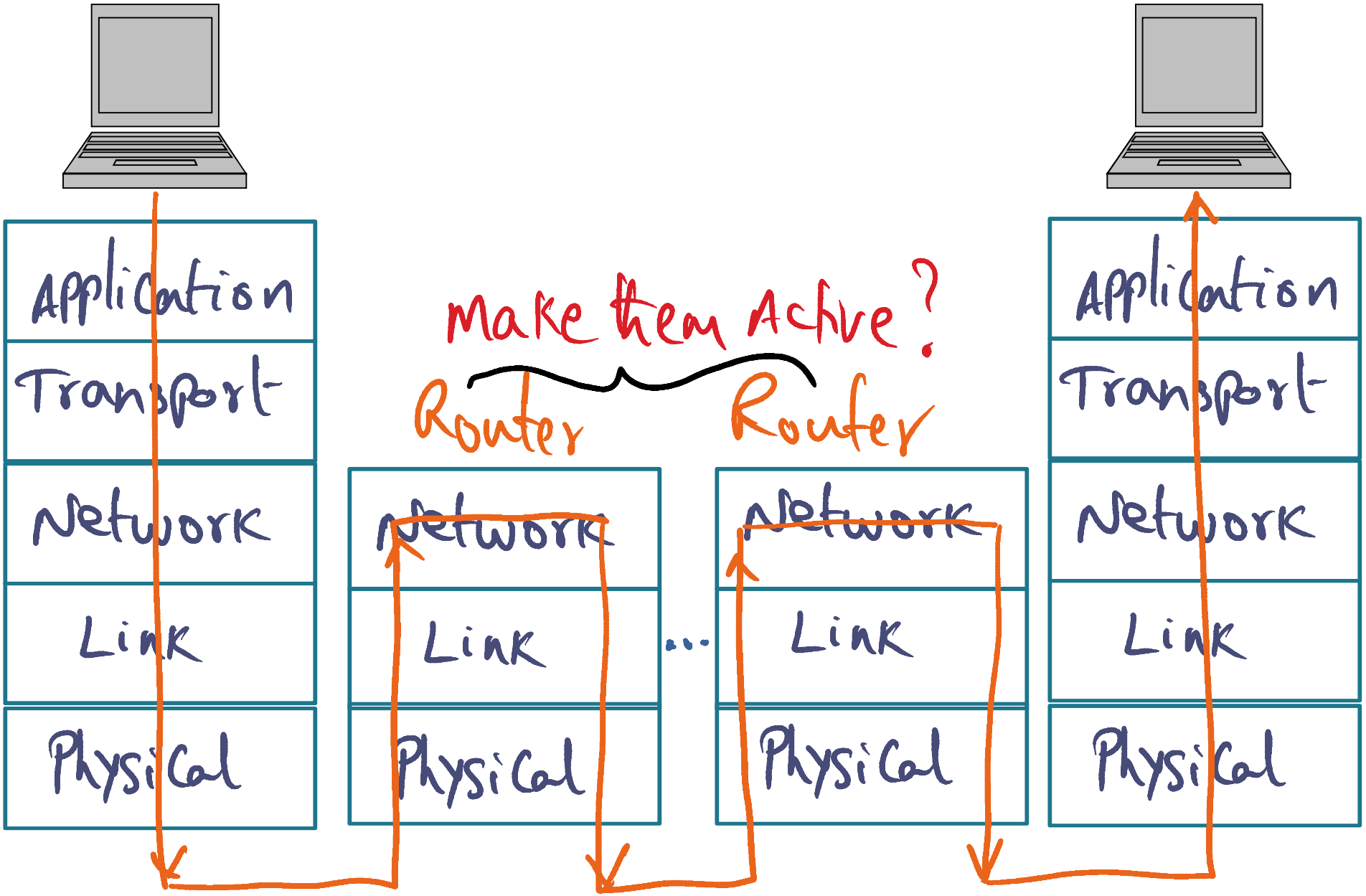
Friday

Synthesizing Network Protocol Stacks

Routing on the Internet

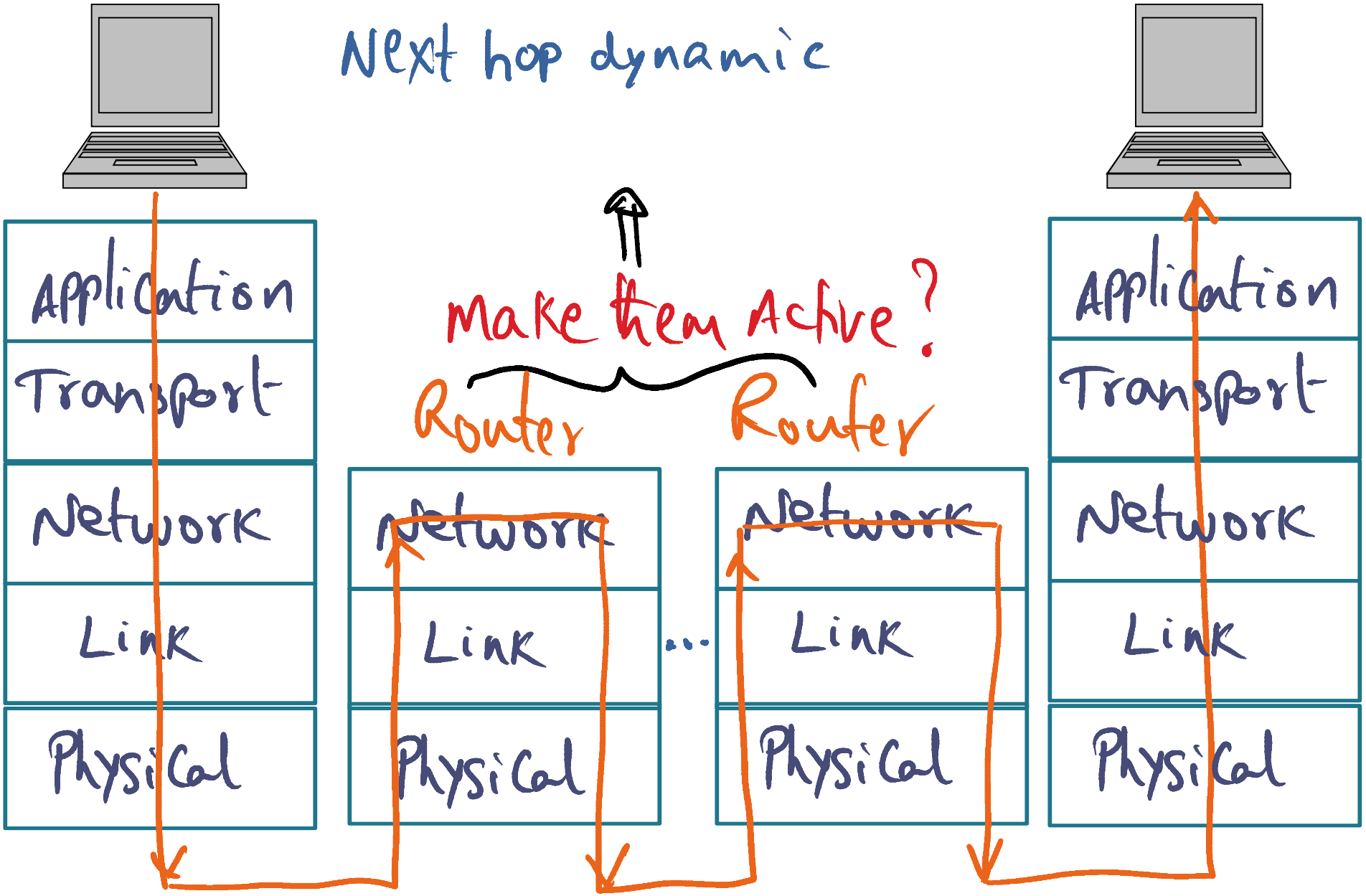


Routing on the Internet

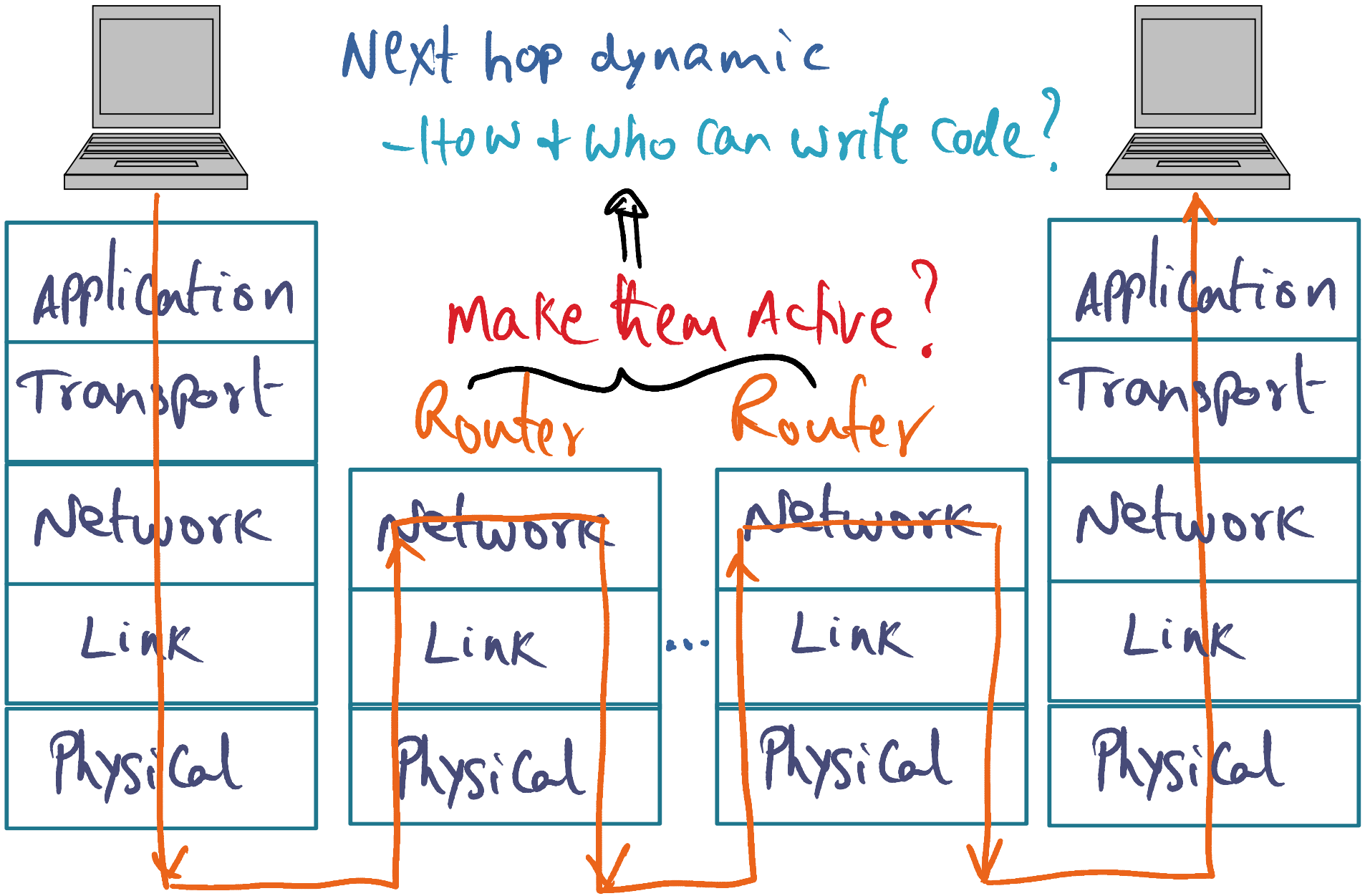


Routing on the Internet

Next hop dynamic



Routing on the Internet

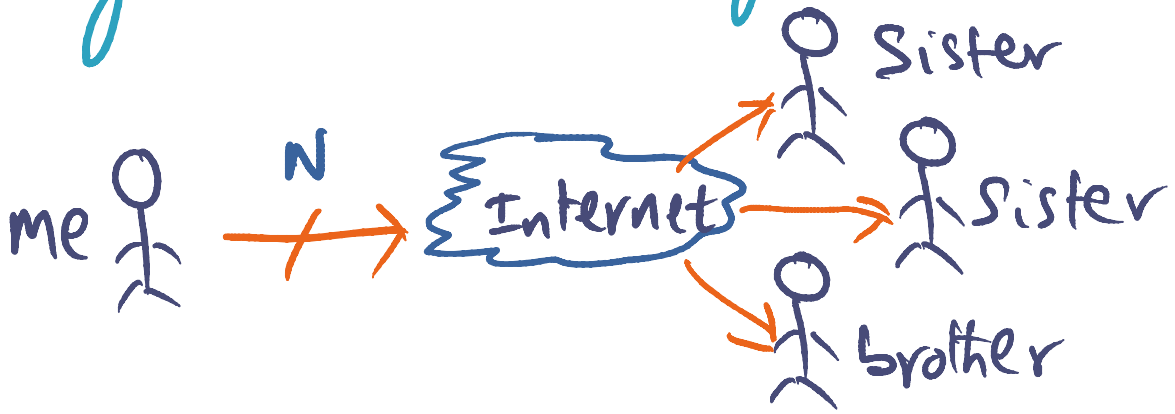


An Example

Me sending Diwali Greetings to my Siblings

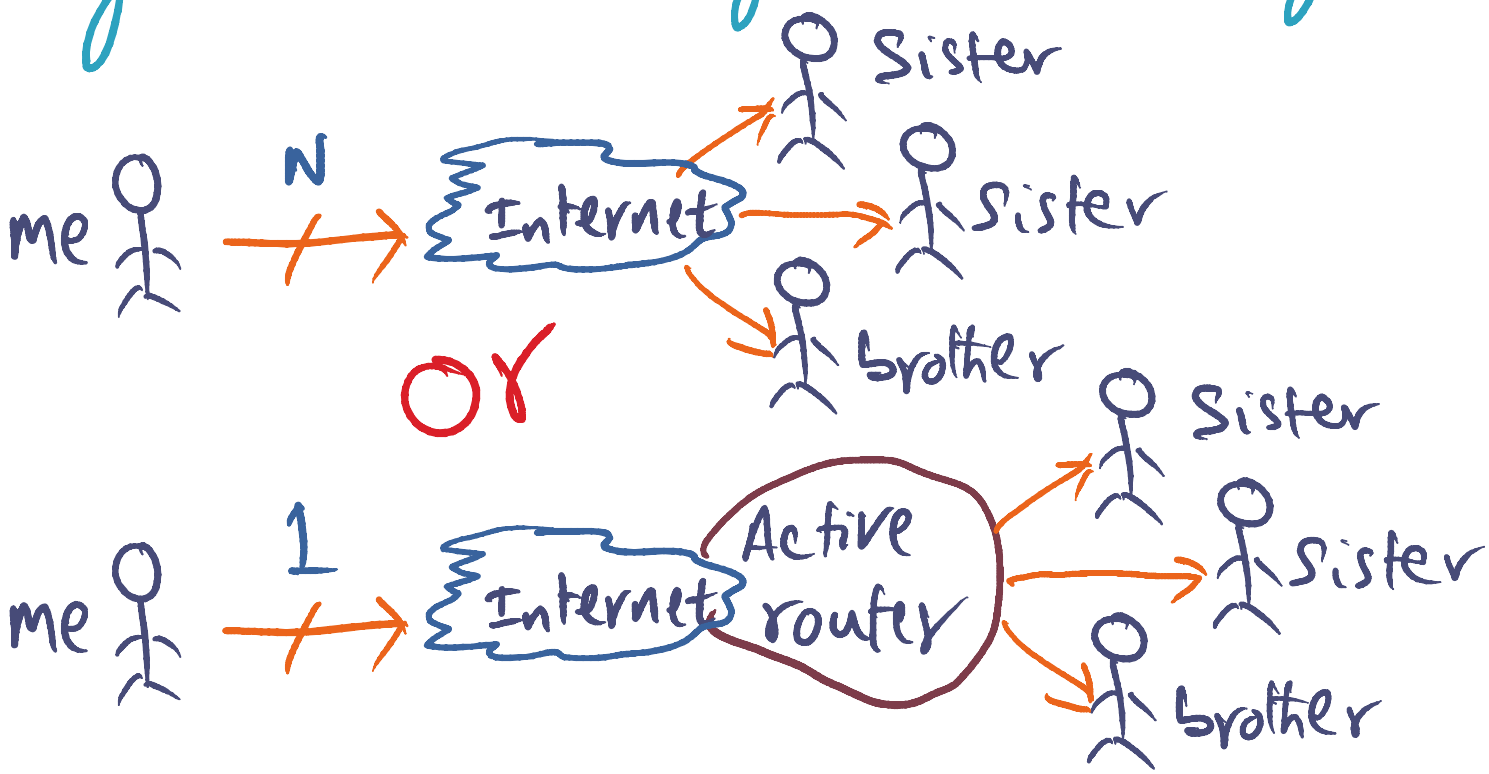
An Example

Me sending Diwali Greetings to my Siblings



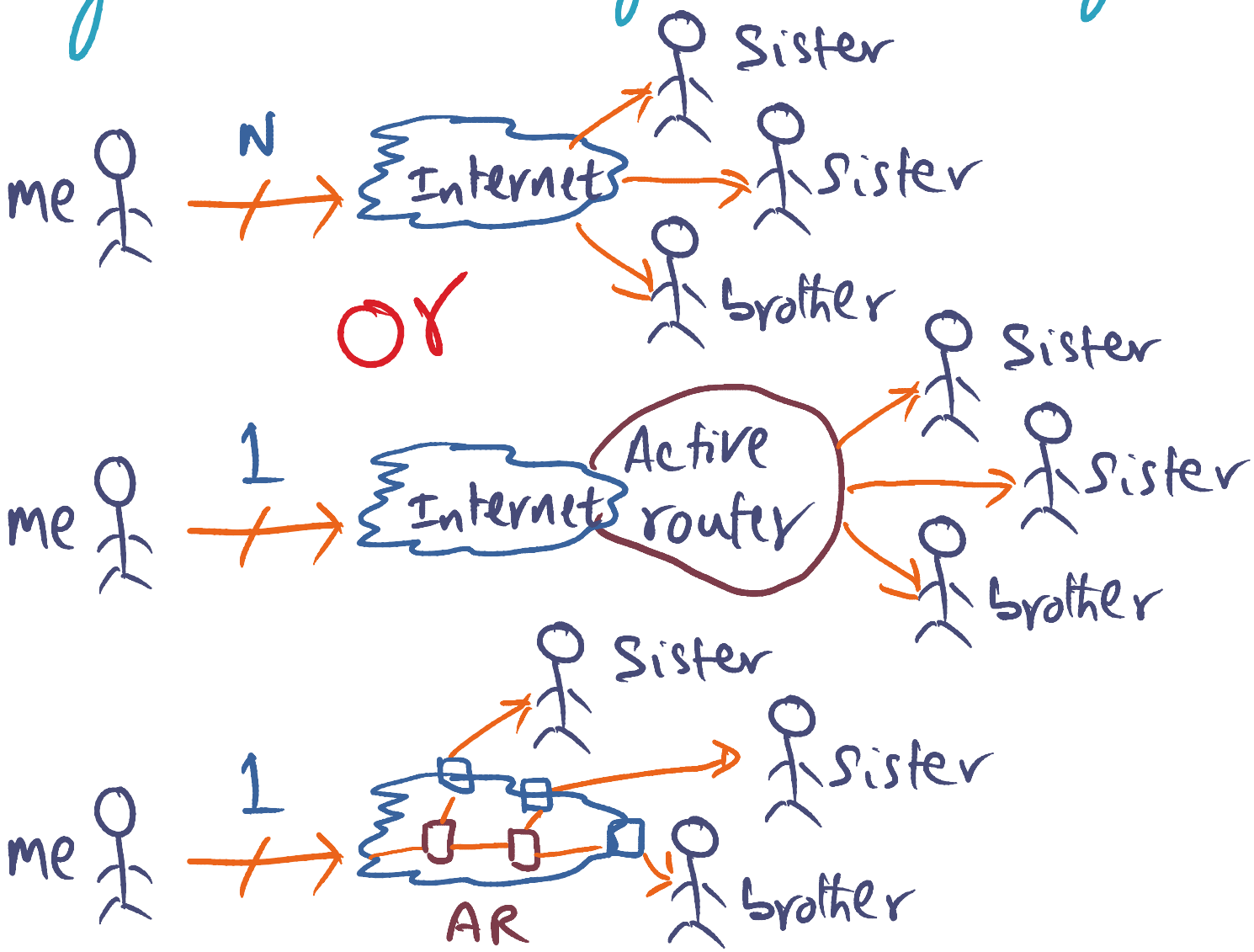
An Example

Me sending Diwali Greetings to my Siblings



An Example

Me sending Diwali Greetings to my Siblings



How to implement the vision?