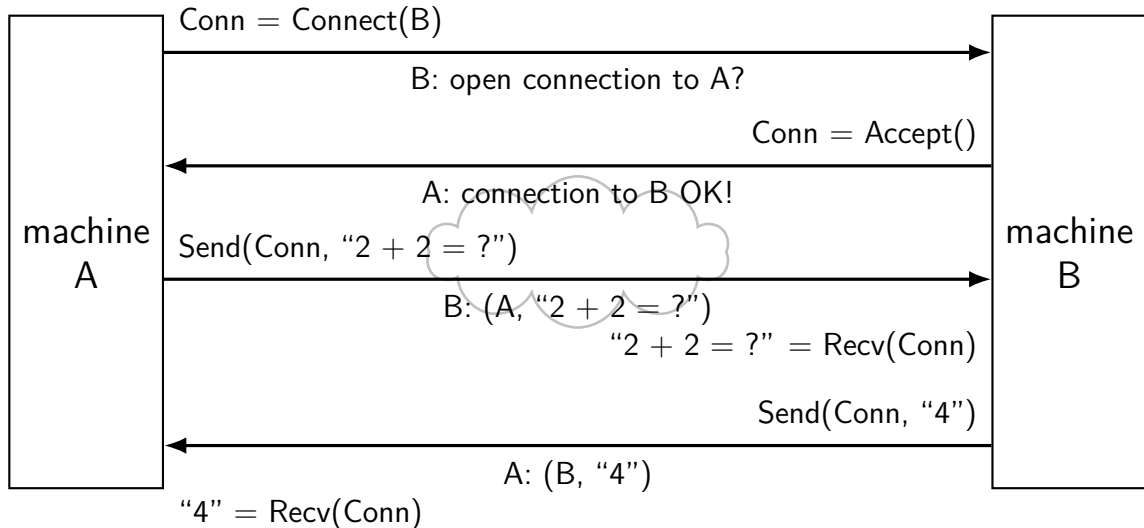


connections

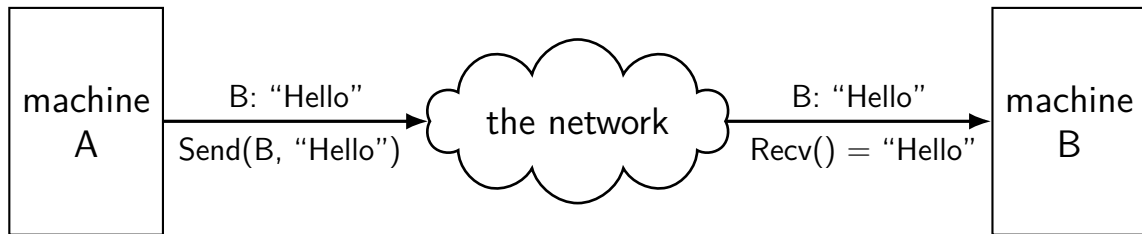
connections: two-way channel for messages

extra operations: connect, accept



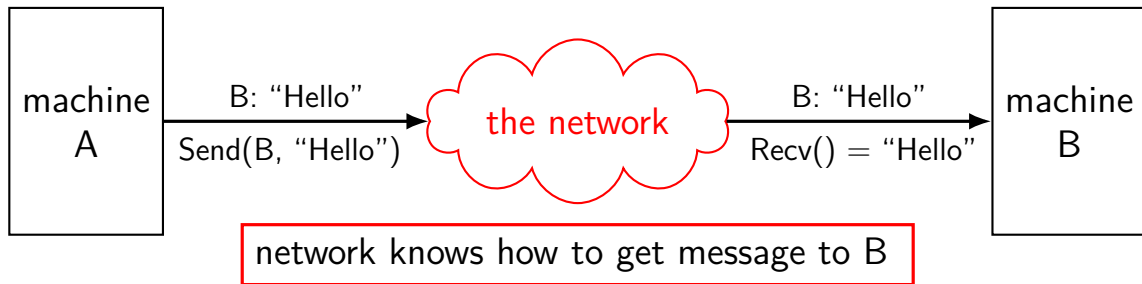
mailbox model

mailbox abstraction: send/receive messages



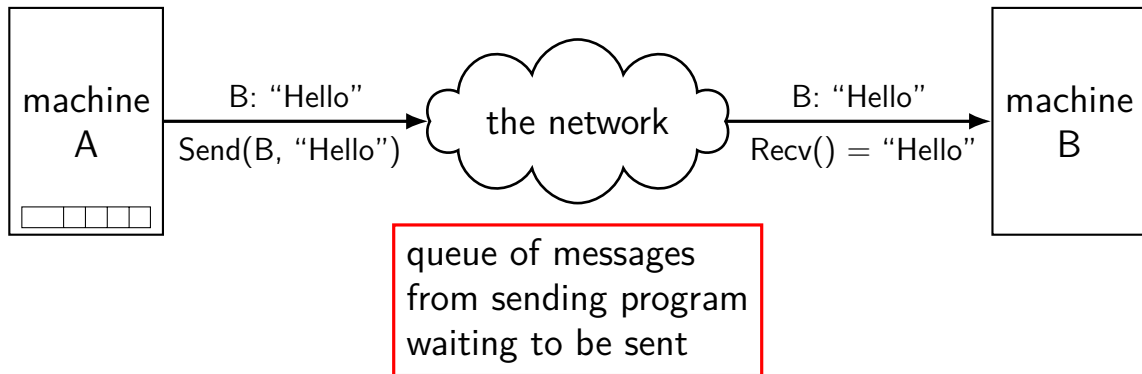
mailbox model

mailbox abstraction: send/receive messages



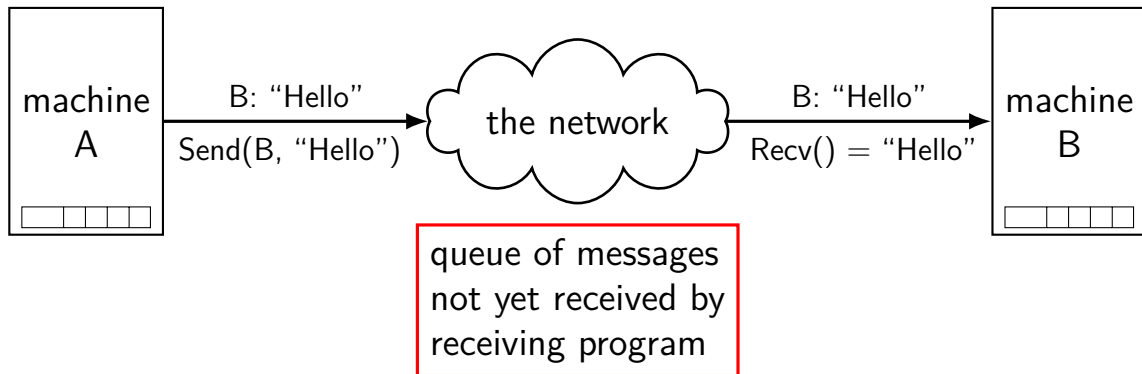
mailbox model

mailbox abstraction: send/receive messages



mailbox model

mailbox abstraction: send/receive messages



connections over mailboxes

real Internet: mailbox-style communication

- send packets to particular mailboxes

- no guarantee on order, when received

sockets implemented on top of this

recall: sockets

open connection then ...

read+write just like a terminal file

doesn't look like individual messages

“connection abstraction”

layers

application	HTTP, SSH, SMTP, ...	application-defined meanings
transport	TCP, UDP, ...	reach correct program, reliability/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

layers

application	HTTP, SSH, SMTP, ...	application-defined meanings
transport	TCP, UDP, ...	reach correct program, reliability/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

network limitations/failures

messages lost

messages delayed/reordered

messages limited in size

messages corrupted

network limitations/failures

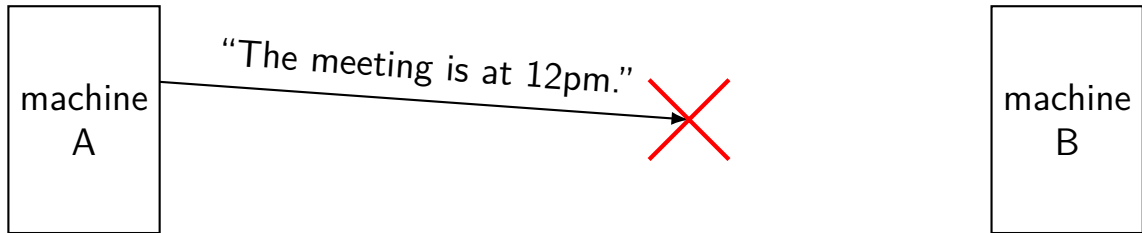
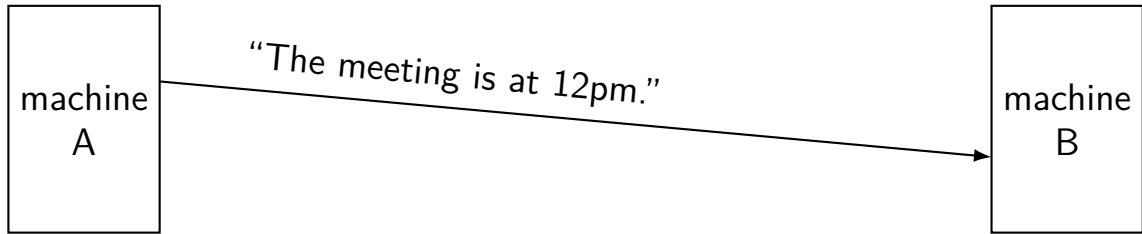
messages lost

messages delayed/reordered

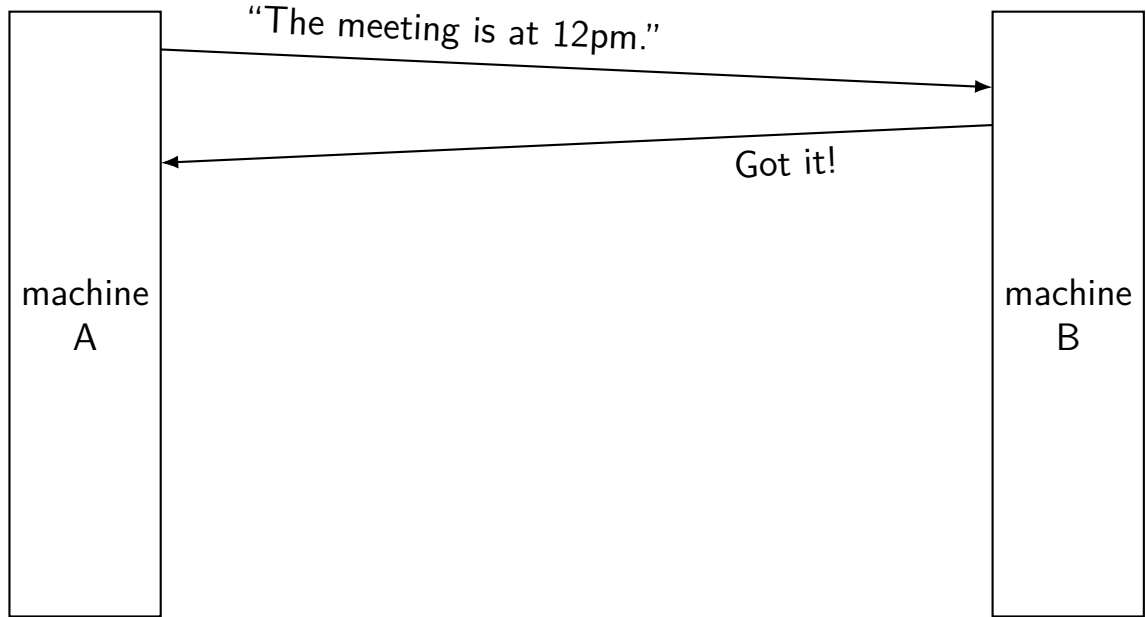
messages limited in size

messages corrupted

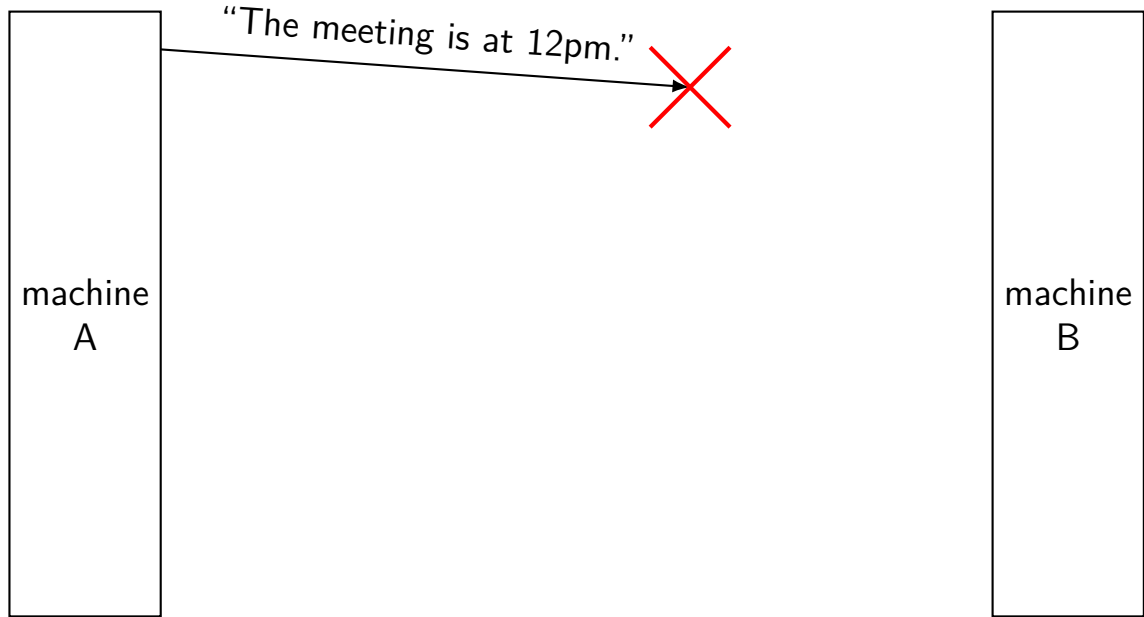
dealing with network message lost



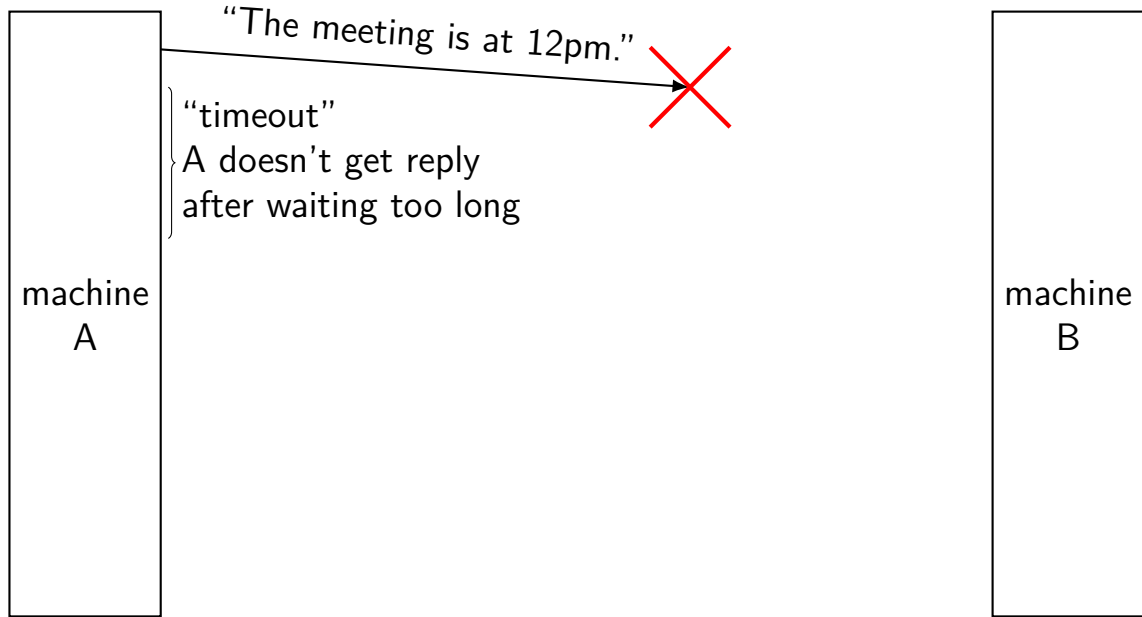
handling lost message: acknowledgements



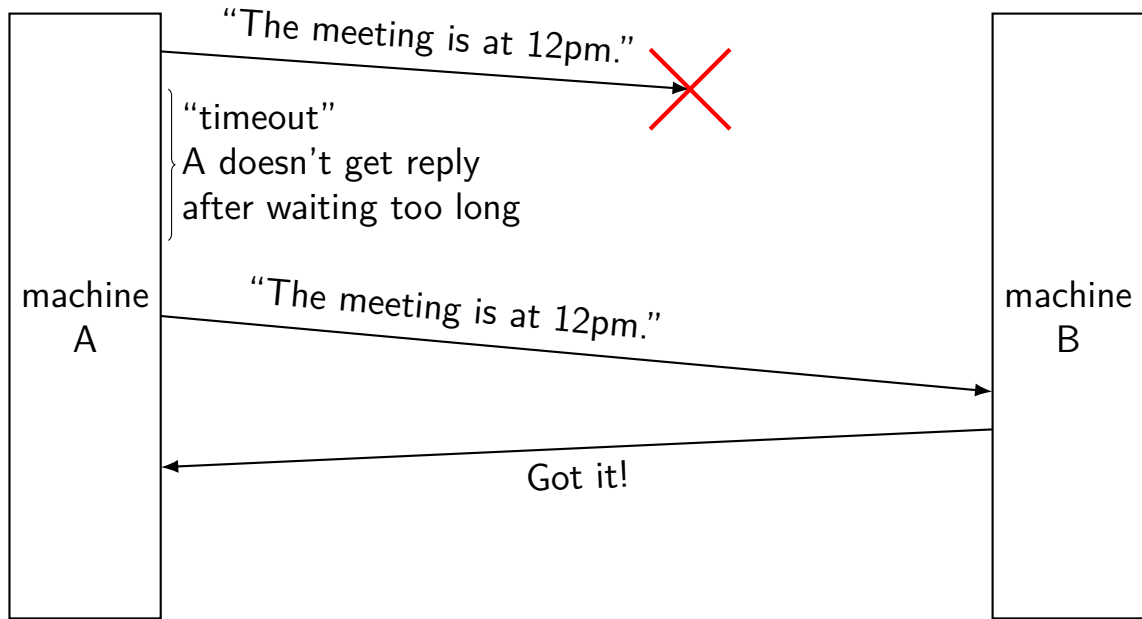
handling lost message



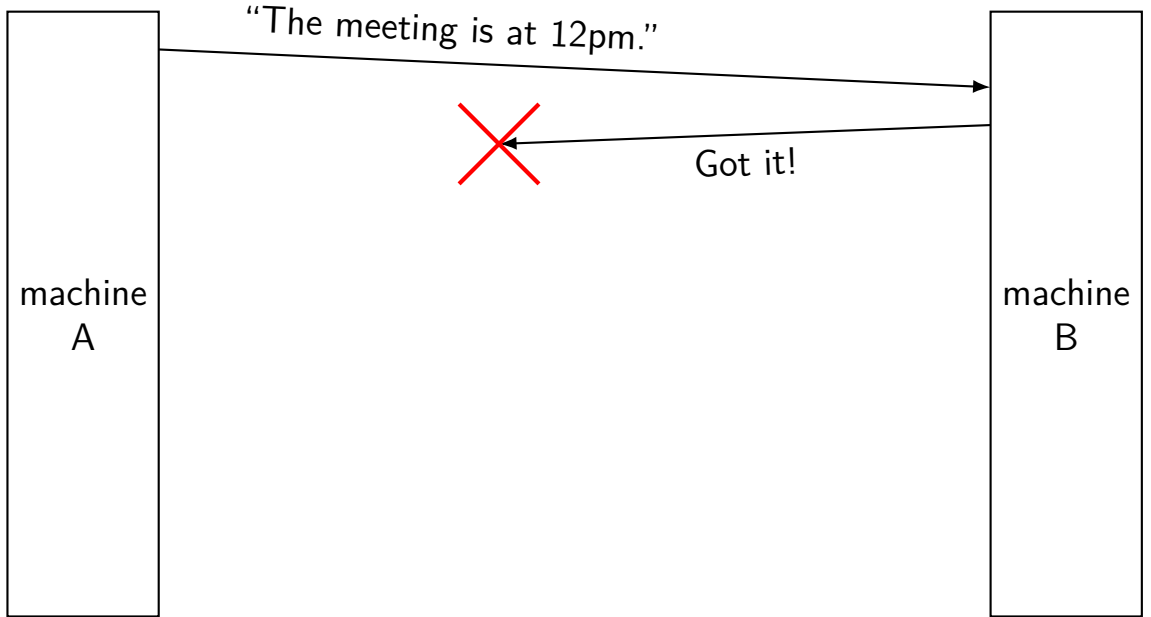
handling lost message



handling lost message

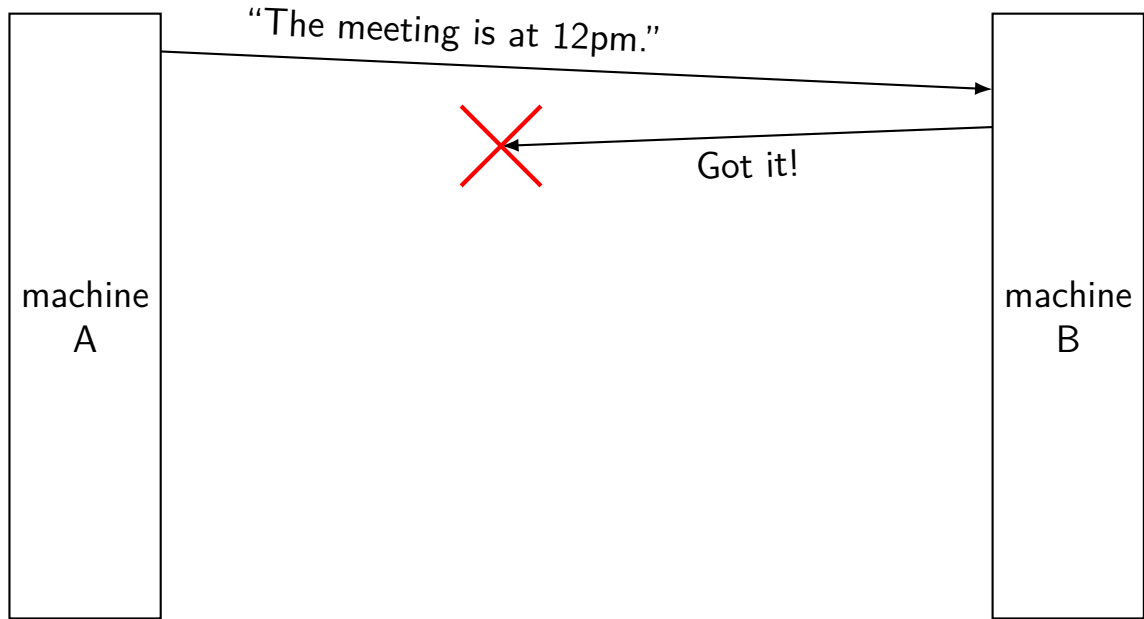


exercise: lost acknowledgement

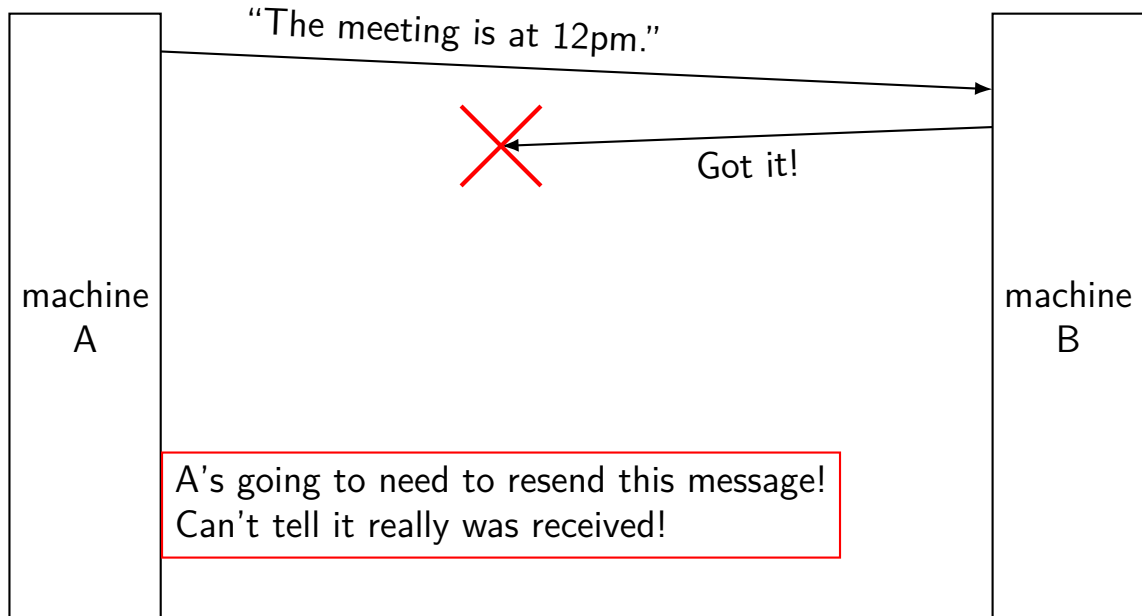


exercise: how to fix this?

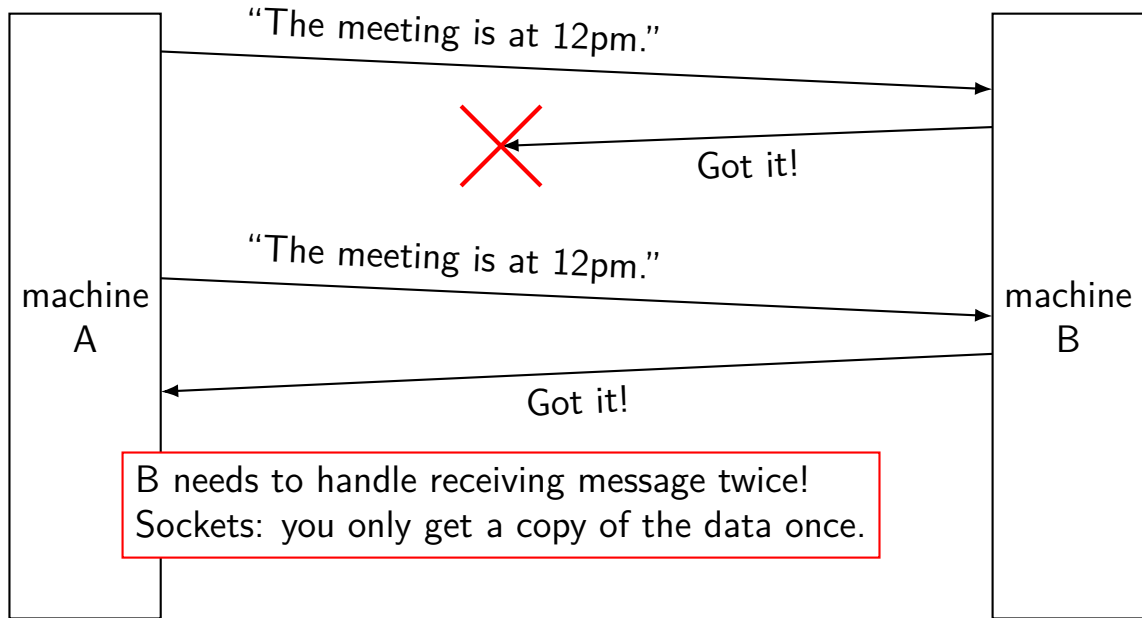
lost acknowledgements



lost acknowledgements



lost acknowledgements



network limitations/failures

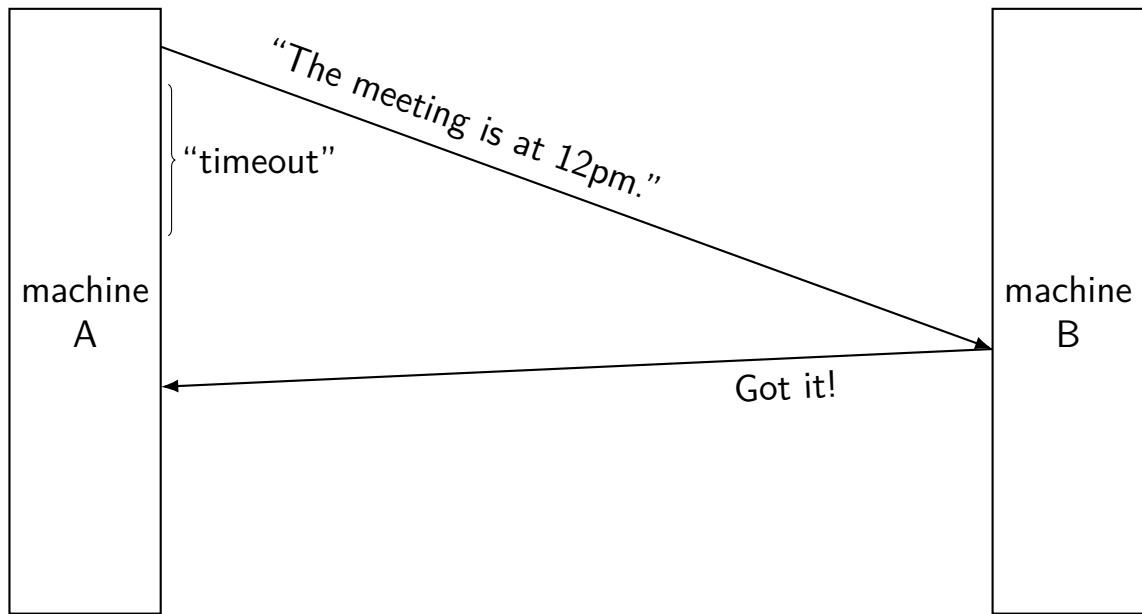
messages lost

messages delayed/reordered

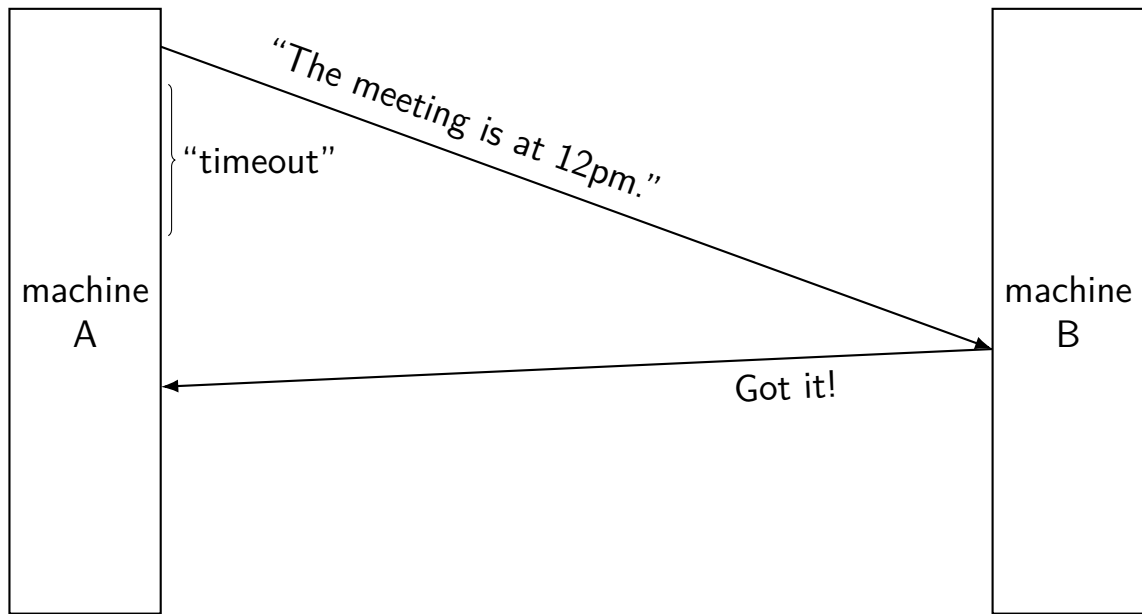
messages limited in size

messages corrupted

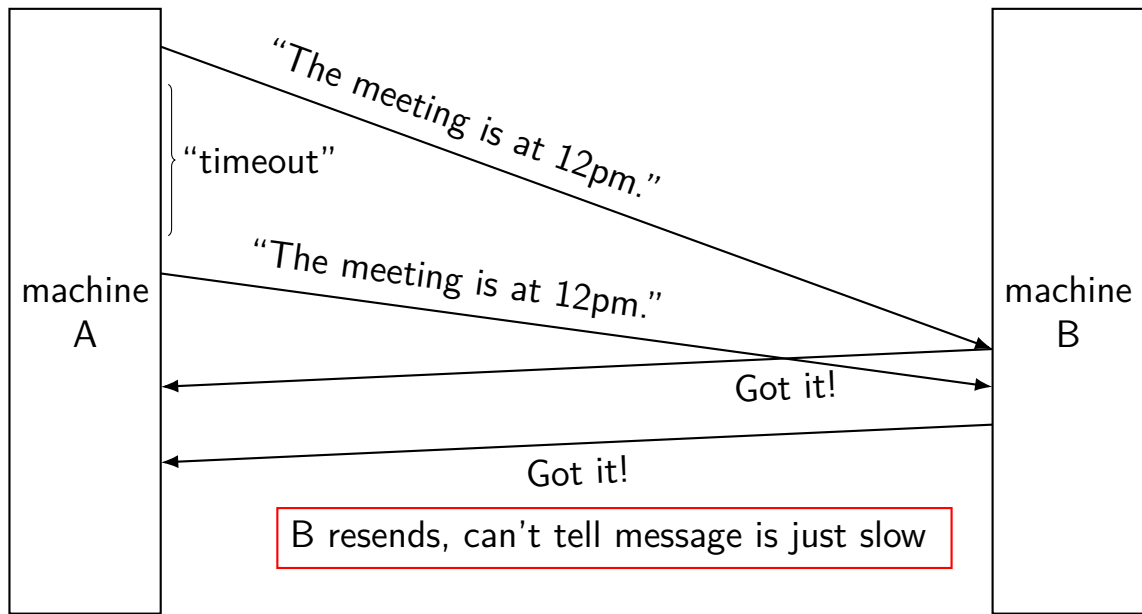
delayed message



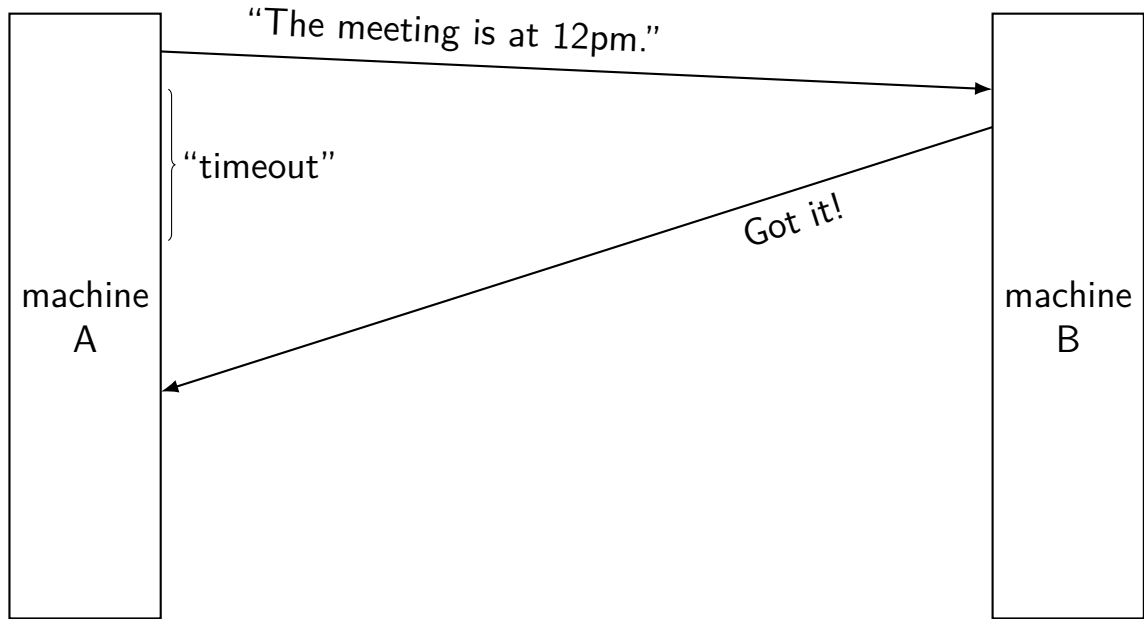
delayed message



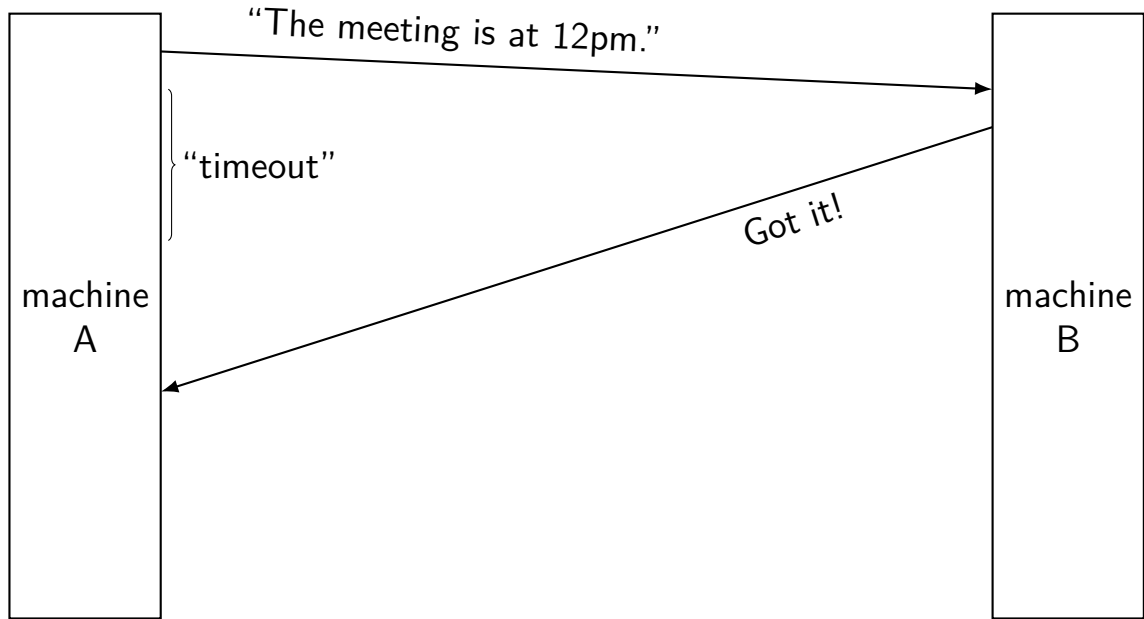
delayed message



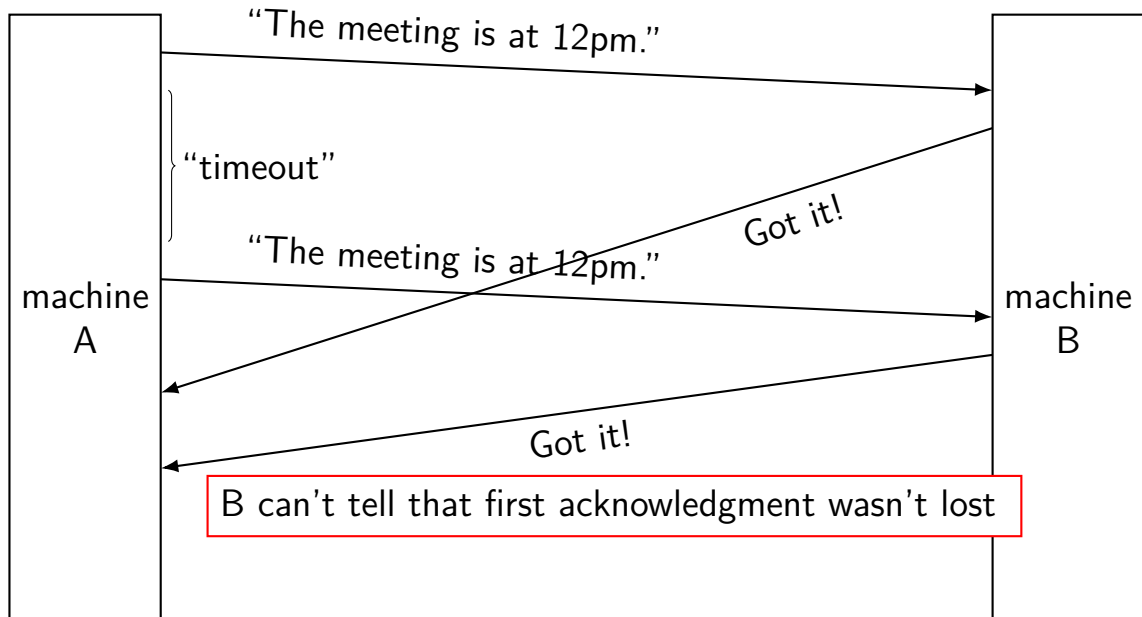
delayed acknowledgements



delayed acknowledgements



delayed acknowledgements



network limitations/failures

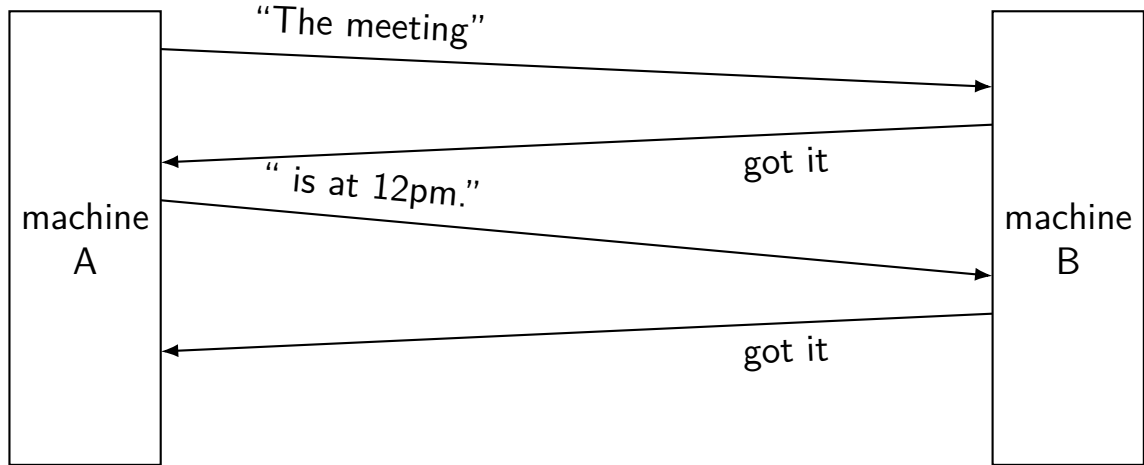
messages lost

messages delayed/reordered

messages limited in size

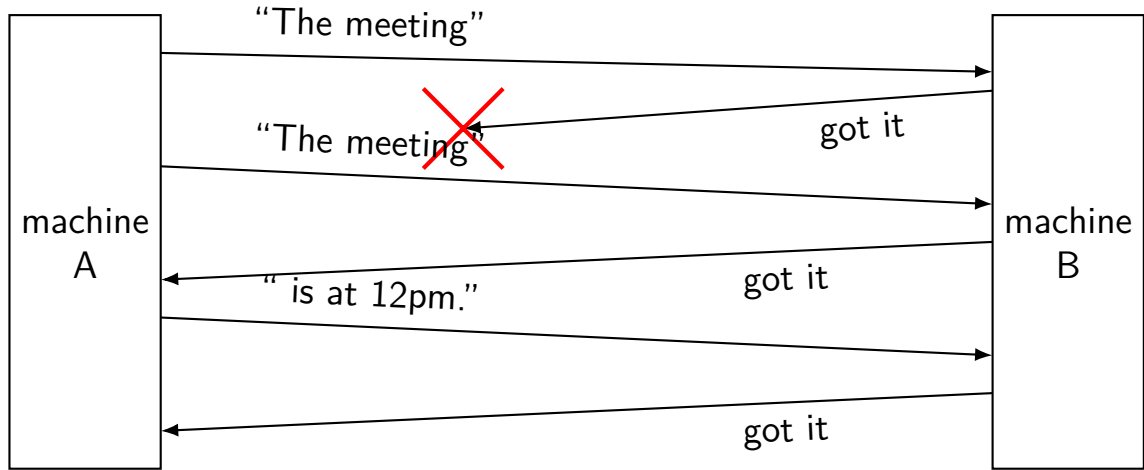
messages corrupted

splitting messages: try 1

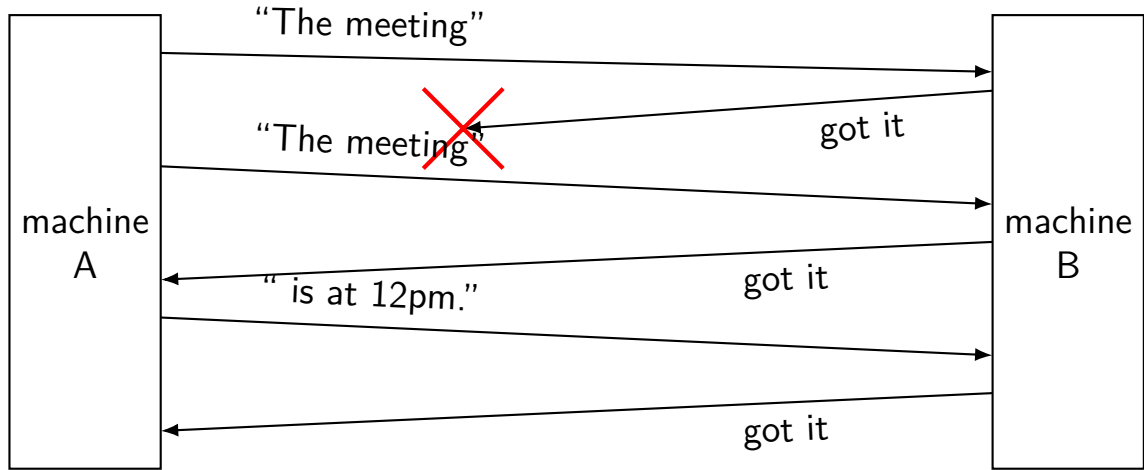


reconstructed message:
The meeting is at 12pm.

splitting messages: try 1 — problem 1



splitting messages: try 1 — problem 1



reconstructed message:

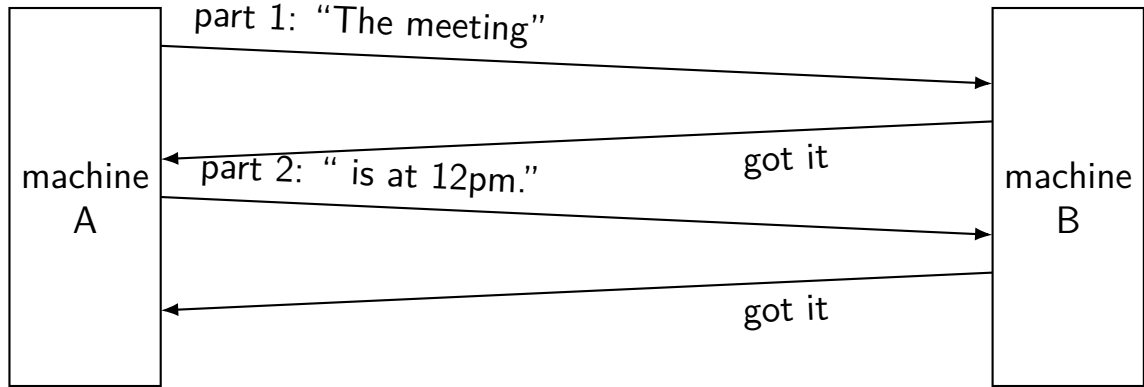
The meetingThe meeting is at 12pm.

exercise: other problems?

other scenarios where we'd also have problems?

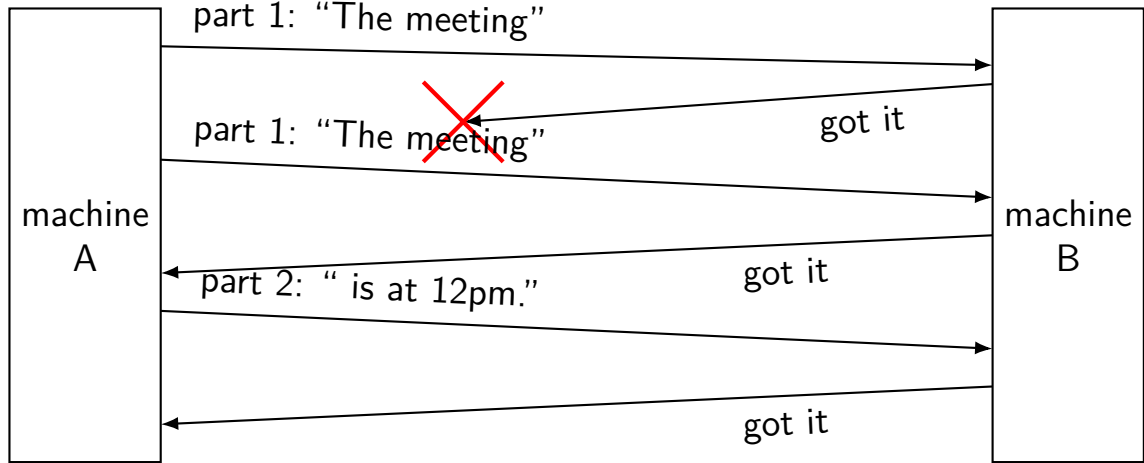
- A. message (instead of acknowledgment) is lost
- B. first message from A is delayed a long time by network
- C. acknowledgment of second message lost instead of first

splitting messages: try 2



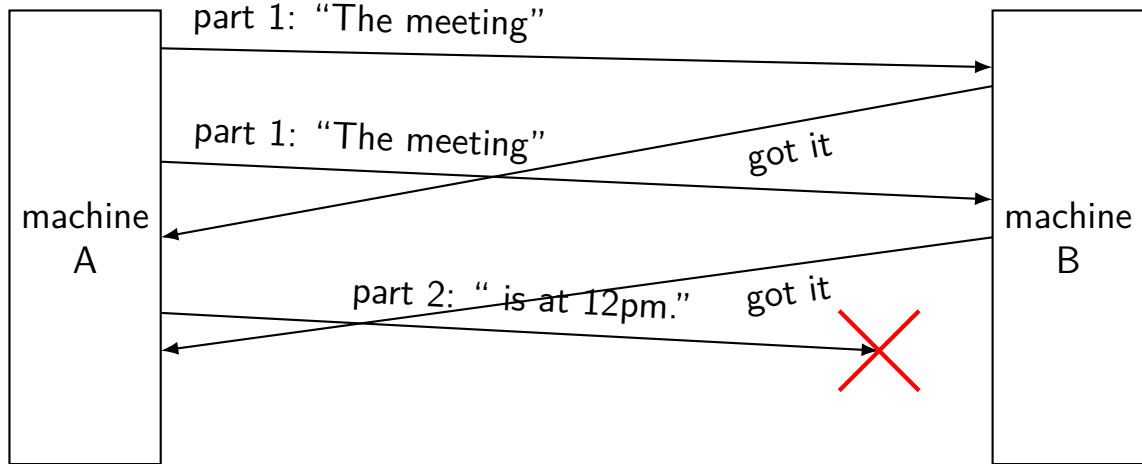
reconstructed message:
The meeting is at 12pm.

splitting messages: try 2 — missed ack



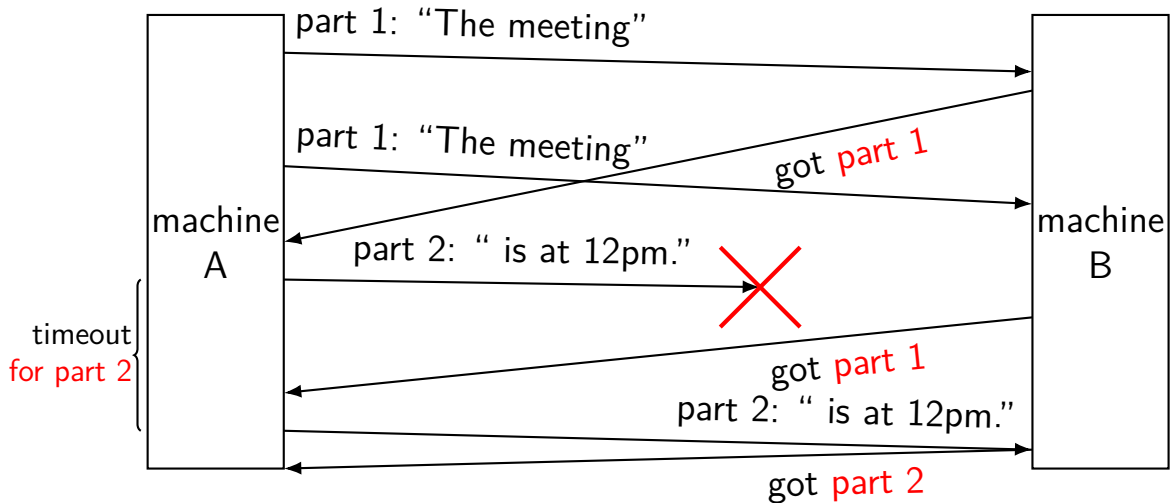
reconstructed message:
The meeting is at 12pm.

splitting messages: try 2 — problem



A thinks: part 1 + part 2 acknowledged!

splitting messages: version 3



network limitations/failures

messages lost

messages delayed/reordered

messages limited in size

messages corrupted

message corrupted

instead of sending “message”

say $\text{Hash}(\text{“message”}) = 0x\text{ABCDEF12}$

then send “0xABCDEF12,message”

when receiving, recompute hash

pretend message lost if does not match

“checksum”

these hashes commonly called “checksums”

in UDP/TCP, produced by adding message parts together (with wraparound)

going faster

so far: send one message, get acknowledgments

pretty slow

instead, can send a bunch of parts and get them acknowledged together

need to do *congestion control* to avoid overloading network

layers

application	HTTP, SSH, SMTP, ...	application-defined meanings
transport	TCP, UDP, ...	reach correct program, reliability/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

more than four layers?

sometimes more layers above 'application'

e.g. HTTPS:

HTTP (app layer) on TLS (another app layer) on TCP (network) on ...

e.g. DNS over HTTPS:

DNS (app layer) on HTTP on on TLS on TCP on ...

e.g. SFTP:

SFTP (app layer??) on SSH (another app layer) on TCP on ...

e.g. HTTP over OpenVPN:

HTTP on TCP on IP on OpenVPN on UDP on different IP on ...

names and addresses

name	address
logical identifier	location/how to locate
variable counter	memory address 0x7FFF9430
DNS name www.virginia.edu	IPv4 address 128.143.22.36
DNS name mail.google.com	IPv4 address 216.58.217.69
DNS name mail.google.com	IPv6 address 2607:f8b0:4004:80b
DNS name reiss-t3620.cs.virginia.edu	IPv4 address 128.143.67.91
DNS name reiss-t3620.cs.virginia.edu	MAC address 18:66:da:2e:7f
service name https	port number 443
service name ssh	port number 22

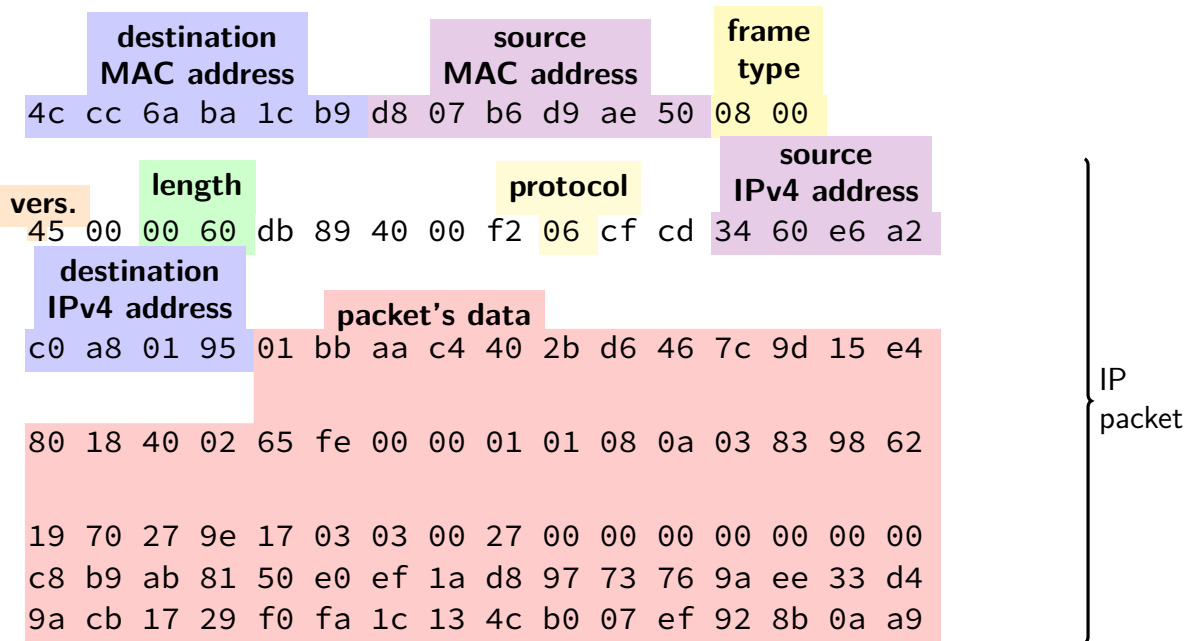
layers

application	HTTP, SSH, SMTP, ...	application-defined meanings
transport	TCP, UDP, ...	reach correct program, reliability/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

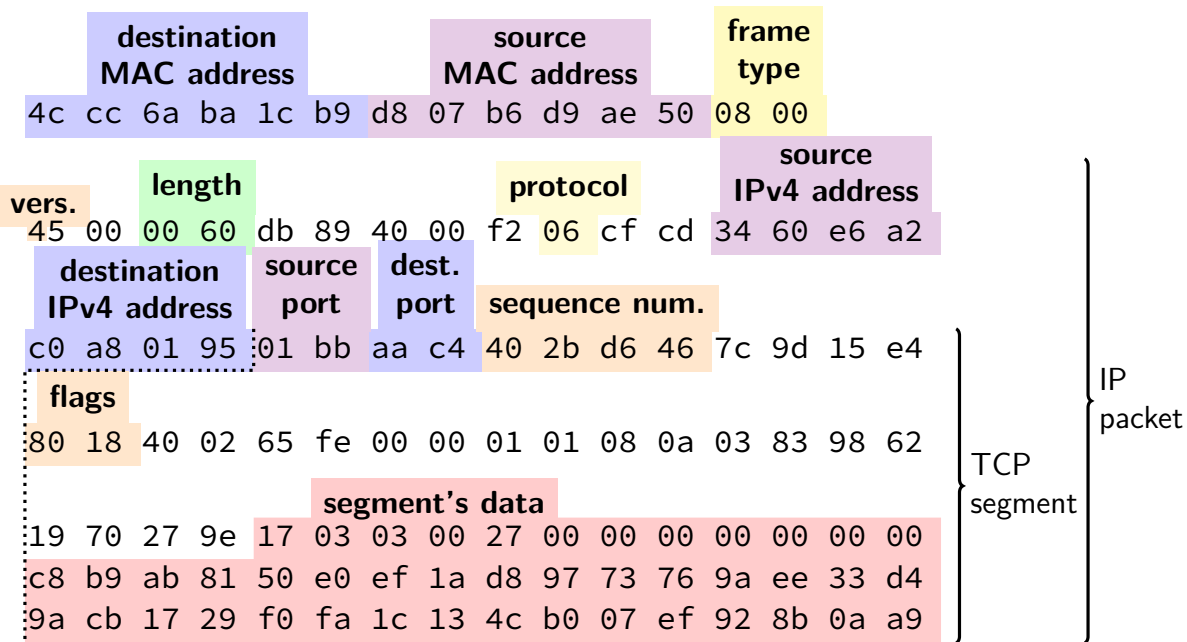
an Ethernet frame

destination MAC address						source MAC address						frame type			
4c	cc	6a	ba	1c	b9	d8	07	b6	d9	ae	50	08	00		
frame's data															
45	00	00	60	db	89	40	00	f2	06	cf	cd	34	60	e6	a2
c0	a8	01	95	01	bb	aa	c4	40	2b	d6	46	7c	9d	15	e4
80	18	40	02	65	fe	00	00	01	01	08	0a	03	83	98	62
19	70	27	9e	17	03	03	00	27	00	00	00	00	00	00	00
c8	b9	ab	81	50	e0	ef	1a	d8	97	73	76	9a	ee	33	d4
9a	cb	17	29	f0	fa	1c	13	4c	b0	07	ef	92	8b	0a	a9

an Ethernet frame



an Ethernet frame



the link layer

Ethernet, Wi-Fi, Bluetooth, DOCSIS (cable modems), ...

allows send/recv messages to machines on “same” network segment

- typically: wireless range+channel or connected to a single switch/router
- could be larger (if *bridging* multiple network segments)
- could be smaller (switch/router uses “virtual LANs”)

typically: source+destination specified with MAC addresses

- MAC = media access control

- usually manufacturer assigned / hard-coded into device
- unique address per port/wifi transmitter/etc.

can specify destination of “anyone” (called *broadcast*)

messages usually called “frames”

link layer quality of service

if frame gets...

event	on Ethernet	on WiFi
collides with another	detected + may resend	resend
not received	lose silently	resent
header corrupted	usually discard silently	usually resend
data corrupted	usually discard silently	usually resend
too long	not allowed to send	not allowed to send
reordered (v. other messages)	received out of order	received out of order
destination unknown	lose silently	usually resend??
too much being sent	discard excess?	discard excess?

layers

application	HTTP, SSH, SMTP, ...	application-defined meanings
transport	TCP, UDP, ...	reach correct program, reliability/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

the network layer

the Internet Protocol (IP) version 4 or version 6

there are also others, but quite uncommon today

allows send messages to/recv messages from other networks

“internetwork”

messages usually called “packets”

network layer quality of service

if packet ...

event

on IPv4/v6

collides with another

out of scope — handled by link layer

not received

lost silently

header corrupted

usually discarded silently

data corrupted

received corrupted

too long

dropped with notice or “fragmented” + recombined

reordered (v. other messages)

received out of order

destination unknown

usually dropped with notice

too much being sent

discard excess

network layer quality of service

if packet ...

event

on IPv4/v6

collides with another

out of scope — handled by link layer

not received

lost silently

header corrupted

usually discarded silently

data corrupted

received corrupted

too long

dropped with notice or “fragmented” + recombined

reordered (v. other messages)

received out of order

destination unknown

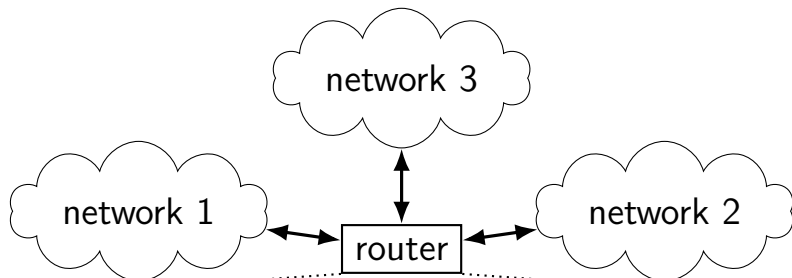
usually dropped with notice

too much being sent

discard excess

includes dropped by link layer
(e.g. if detected corrupted there)

IPv4 addresses and routing tables



if I receive data for...	send it to...
128.143.0.0—128.143.255.255	network 1
192.107.102.0—192.107.102.255	network 1
...	...
4.0.0.0—7.255.255.255	network 2
64.8.0.0—64.15.255.255	network 2
...	...
anything else	network 3

layers

application	HTTP, SSH, SMTP, ...	application-defined meanings
transport	TCP, UDP, ...	reach correct program, reliability/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

port numbers

we run multiple programs on a machine

IP addresses identifying machine — not enough

port numbers

we run multiple programs on a machine

IP addresses identifying machine — not enough

so, add 16-bit *port numbers*

think: multiple PO boxes at address

port numbers

we run multiple programs on a machine

IP addresses identifying machine — not enough

so, add 16-bit *port numbers*

think: multiple PO boxes at address

0–49151: typically assigned for particular services

80 = http, 443 = https, 22 = ssh, ...

49152–65535: allocated on demand

default “return address” for client connecting to server

UDP v TCP

UDP: messages sent to program, but no reliability/streams

- get assigned port number

- SOCK_DGRAM with `socket()` instead of `SOCK_STREAM`

- can `sendto()`/`recvfrom()` multiple other programs with one socket

 - (but don't have to)

- send messages which are limited in size, unreliable

TCP: stream to other program

- need to `bind()` + `listen()` + `accept()` or `connect()` to setup connection

- one socket per connection

- read/write bytes — divided into messages automatically

- reliable — acknowledgments/resending handled for you

connections in TCP/IP

connection identified by *5-tuple*

used by OS to lookup “where is the socket?”

(protocol=TCP/UDP, local IP addr., local port, remote IP addr., remote port)

local IP address, port number can be set with `bind()` function

typically always done for servers, not done for clients

system will choose default if you don't

connections on my desktop

```
cr4bd@reiss-t3620>/u/cr4bd
```

```
$ netstat --inet --inet6 --numeric
```

```
Active Internet connections (w/o servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	128.143.67.91:49202	128.143.63.34:22	ESTABLISHED
tcp	0	0	128.143.67.91:803	128.143.67.236:2049	ESTABLISHED
tcp	0	0	128.143.67.91:50292	128.143.67.226:22	TIME_WAIT
tcp	0	0	128.143.67.91:54722	128.143.67.236:2049	TIME_WAIT
tcp	0	0	128.143.67.91:52002	128.143.67.236:111	TIME_WAIT
tcp	0	0	128.143.67.91:732	128.143.67.236:63439	TIME_WAIT
tcp	0	0	128.143.67.91:40664	128.143.67.236:2049	TIME_WAIT
tcp	0	0	128.143.67.91:54098	128.143.67.236:111	TIME_WAIT
tcp	0	0	128.143.67.91:49302	128.143.67.236:63439	TIME_WAIT
tcp	0	0	128.143.67.91:50236	128.143.67.236:111	TIME_WAIT
tcp	0	0	128.143.67.91:22	172.27.98.20:49566	ESTABLISHED
tcp	0	0	128.143.67.91:51000	128.143.67.236:111	TIME_WAIT
tcp	0	0	127.0.0.1:50438	127.0.0.1:631	ESTABLISHED
tcp	0	0	127.0.0.1:631	127.0.0.1:50438	ESTABLISHED

non-connection sockets

TCP servers waiting for connections +
UDP sockets with no particular remote host

Linux: OS keeps 5-tuple with “wildcard” remote address

“listening” sockets on my desktop

```
cr4bd@reiss-t3620>/u/cr4bd
```

```
$ netstat --inet --inet6 --numeric --listen
```

```
Active Internet connections (only servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	State
tcp	0	0	127.0.0.1:38537	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:36777	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:41099	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:45291	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:51949	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:41071	0.0.0.0:*	LISTEN
tcp	0	0	0.0.0.0:111	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:32881	0.0.0.0:*	LISTEN
tcp	0	0	127.0.0.1:38673	0.0.0.0:*	LISTEN
....					
tcp6	0	0	:::42689	:::*	LISTEN
udp	0	0	128.143.67.91:60001	0.0.0.0:*	
udp	0	0	128.143.67.91:60002	0.0.0.0:*	
...					
udp6	0	0	:::59938	:::*	

URL / URIs

Uniform Resource Locators (URL)

tells how to find “resource” on network

Uniform Resource Identifiers

superset of URLs

URI examples

`https://kytos02.cs.virginia.edu:443/cs3130-spring2023/
quizzes/quiz.php?qid=02#q2`

`https://kytos02.cs.virginia.edu/cs3130-spring2023/
quizzes/quiz.php?qid=02`

`https://www.cs.virginia.edu/`

`sftp://cr4bd@portal.cs.virginia.edu/u/cr4bd/file.txt`

`tel:+1-434-982-2200`

URI generally

scheme://authority/path?query#fragment

scheme: — what protocol

//authority/

authorirty = user@host:port OR host:port OR user@host OR host

path

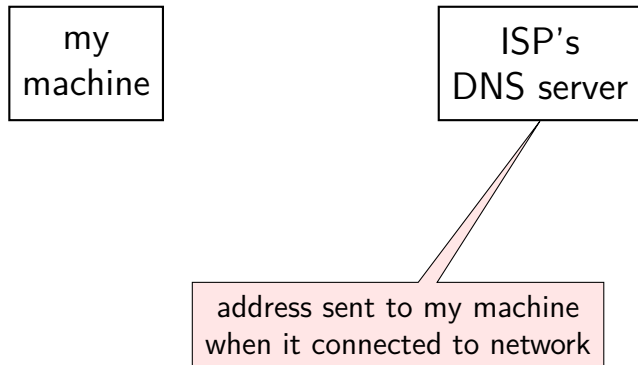
which resource

?query — usually key/value pairs

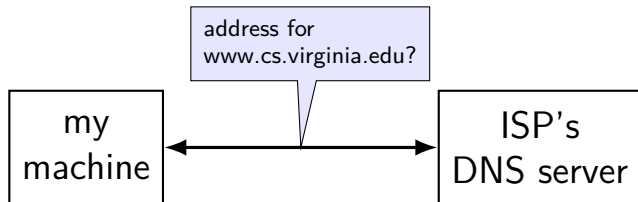
#fragment — place in resource

most components (sometimes) optional

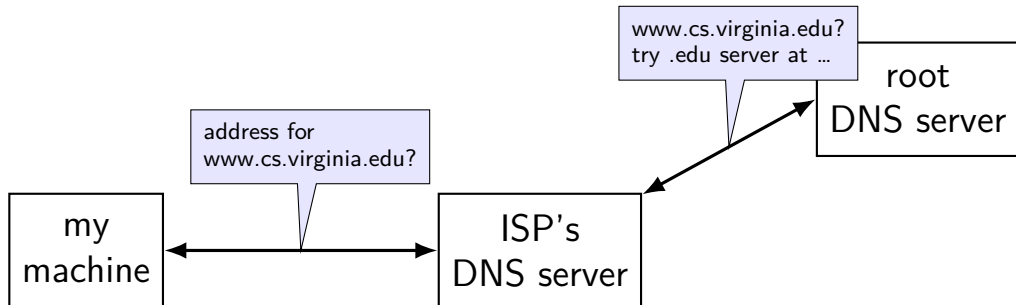
DNS: distributed database



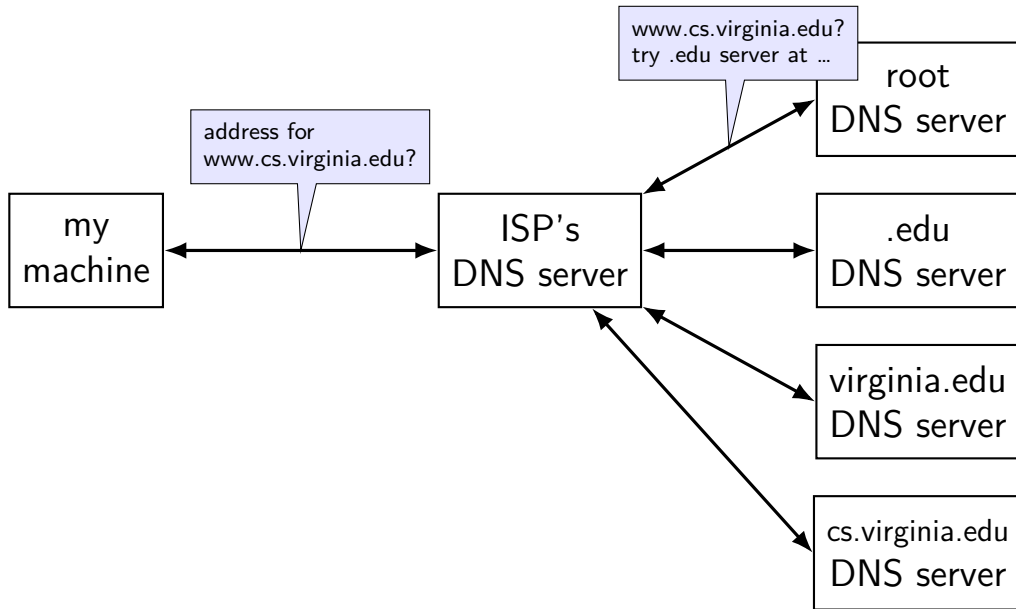
DNS: distributed database



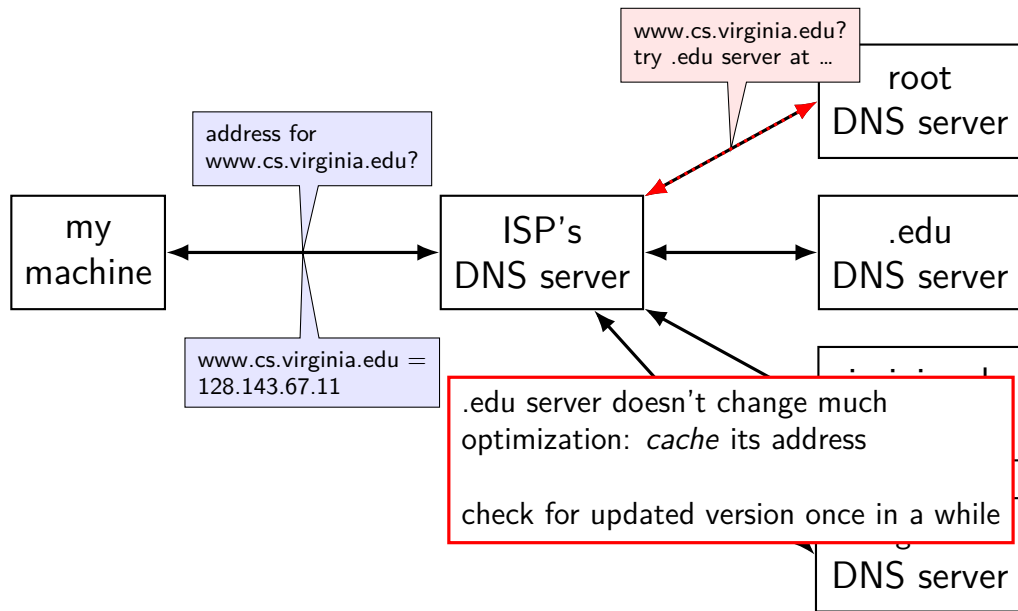
DNS: distributed database



DNS: distributed database



DNS: distributed database



autoconfiguration

problem: how does my machine get IP address

otherwise:

- have sysadmin type one in?

- just choose one?

- ask someone on local network to assign it

autoconfiguration

problem: how does my machine get IP address

otherwise:

- have sysadmin type one in?

- just choose one?

- ask someone on local network to assign it

DHCP high-level

protocol done over UDP

but since we don't have IP address yet, use 0.0.0.0

and since we don't know server address, use 255.255.255.255
= “everyone on the local network”

local server replies to request with address + time limit

backup slides