last time (1)

```
multi-level page tables
```

split virtual page number into parts first part: index in 1st level table 1st level table points to 2nd level table (instead of data) second part: index in 2nd level table

page table permission bits

protecting OS memory but making it accessible without changing PTBR disabling writes for safe sharing

last time (2)

allocate-on-demand

don't tell processor about everything "allocated" to program fixup disagreement on page fault (instead of crashing program)

copy-on-write

tell processor: this is read-only make a copy and fixup disagreement on protection fault

anonymous feedback (1)

"In OH, some of your TAs are incredibly unhelpful. It is clear that they prioritize their friends over other students who need help. This is not the case for all of them, but when it happens it is incredibly frustration to be waiting and not get help because they are helping in order of who they know and not when students arrived."

anonymous feedback (2)

"the amount of piazza questions for this HW and the overwhelming amount of people at OH indicates that the topic was not taught well and the instructions are unclear"

selected common confusion on assignment

```
translate \approx processor's memory lookup can follow lookup diagram from slides (1- or 2-level)
```

```
page_allocate \approx OS's allocation-on-demand allocate things that translate would find missing allocates both page tables and the 'data' they point to assignment originally wasn't explicit about this, but translate() can't work if you don't
```

needs to handle initializing page tables (so translate knows there's nothing there yet)

```
how big are virtual addresses (the part used for translation)? based on page table sizes not all of size_t used
```

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused page table base register 0x20; translate virtual address 0x131

physical bytes addresses_	physical bytes addresses
addresses	addresses
0x00-3 00 11 22 33	0x20-3 00 91 72 13
0x04-744 55 66 77	0x24-7 D4 F5 36 07
0x08-B88 99 AA BE	0x28-B89 9A AB BC
0x0C-FCC DD EE FF	0×2C-FCD DE EF F0
0x10-3 1A 2A 3A 4A	0×30-3BA 0A BA 0A
0x14-7 1B 2B 3B 4B	0×34-7 DB 0B DB 0B
0x18-B1C 2C 3C 4C	0×38-BEC 0C EC 0C
0x1C-F1C 2C 3C 4C	0x3C-FFC 0C FC 0C

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused page table base register 0x20; translate virtual address 0x131

```
physical bytes
                                                   0 \times 131 = 1 \quad 0011 \quad 0001
addresses
                                                   0x20 + 4 \times 1 = 0x24
0 \times 00 - 3 | 00 \ 11 \ 22 \ 33
                           0x20-3|00 91 72 13
                                                   PTE 1 value:
0 \times 04 - 7 | 44 55 66 77
                           0x24-7D4 F5 36 07
                                                   0 \times D4 = 1101 \ 0100
0x08-Bl88 99 AA BB
                           0×28-Bl89 9A AB BC
                                                   PPN 110. valid 1
0x0C-FCC DD EE FF
                           0x2C-FCD DE EF F0
                           0x30-3|BA 0A BA 0A
0 \times 10 - 3 | 1A 2A 3A 4A
                           0x34-7DB 0B DB 0B
0 \times 14 - 7 | 1B 2B 3B 4B
0x18-Bl1C 2C 3C 4C
                           0x38-BIEC 0C EC 0C
                           0x3C-F|FC 0C FC 0C
0x1C-F|1C 2C 3C 4C
```

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused

page table base register 0x20; translate virtual address 0x131

```
physical bytes
                                                  0 \times 131 = 1 \ 0011 \ 0001
addresses
                                                  0x20 + 4 \times 1 = 0x24
0 \times 00 - 3 | 00 \ 11 \ 22 \ 33
                          0x20-3|00 91 72 13
                                                  PTE 1 value:
0 \times 04 - 7 | 44 55 66 77
                          0x24-7D4 F5 36 07
                                                  0 \times D4 = 1101 \ 0100
                          0x28-Bl89 9A AB BC
0x08-Bl88 99 AA BB
                                                  PPN 110. valid 1
0x0C-FCC DD EE FF
                          0x2C-FCD DE EF F0
                                                  PTE 2 addr:
                          0x30-3|BA 0A BA 0A
0 \times 10 - 3 | 1A 2A 3A 4A
                                                   110 000 + 110 \times 1 = 0x36
                          0x34-7DB 0B DB 0B
0 \times 14 - 7 | 1B 2B 3B 4B
                                                   PTE 2 value: 0xDB
0x18-Bl1C 2C 3C 4C
                          0x38-BIEC 0C EC 0C
                          0x3C-F|FC 0C FC 0C
0x1C-F|1C 2C 3C 4C
```

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused page table base register 0x20; translate virtual address 0x131

```
physical bytes
                            physical <sub>bytes</sub>
addresses
                           addresses
                                                    0x20 + 4 \times 1 = 0x24
0x00-3|00 11 22 33
                           0x20-3|00 91 72 13
                                                   PTE 1 value:
                           0x24-7D4 F5 36 07
0 \times 04 - 7 | 44 55 66 77
                                                    0 \times D4 = 1101 \ 0100
0x08-B|88 99 AA BB
                           0x28-B|89 9A AB BC
                                                    PPN 110. valid 1
0x0C-FCC DD EE FF
                           0x2C-FCD DE EF F0
                                                    PTE 2 addr:
0 \times 10 - 3 | 1A 2A 3A 4A
                           0x30-3|BA 0A BA 0A
                                                    110\ 000 + 110 \times 1 = 0x36
                                                    PTE 2 value: 0xDB
0 \times 14 - 7 | 1B 2B 3B 4B
                           0 \times 34 - 7 \mid DB \mid 0B \mid DB \mid 0B
                           0x38-B|EC 0C EC 0C
                                                    PPN 110; valid 1
0x18-Bl1C 2C 3C 4C
                                                    M[110 \ 001 \ (0x31)] = 0x0A
0x1C-F|1C 2C 3C 4C
                           0x3C-FIFC 0C FC 0C
```

 $0 \times 131 = 1 \ 0011 \ 0001$

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused page table base register 0x20; translate virtual address 0x131

```
physical bytes
                            physical <sub>bytes</sub>
addresses
                           addresses
                                                    0x20 + 4 \times 1 = 0x24
0x00-3|00 11 22 33
                           0x20-3|00 91 72 13
                                                   PTE 1 value:
                           0x24-7D4 F5 36 07
0 \times 04 - 7 | 44 55 66 77
                                                    0 \times D4 = 1101 \ 0100
0x08-B|88 99 AA BB
                           0x28-B|89 9A AB BC
                                                    PPN 110. valid 1
0x0C-FCC DD EE FF
                           0x2C-FCD DE EF F0
                                                    PTE 2 addr:
0 \times 10 - 3 | 1A 2A 3A 4A
                           0x30-3|BA 0A BA 0A
                                                    110 \ 000 + 110 \times 1 = 0x36
                                                    PTE 2 value: 0xDB
0 \times 14 - 7 | 1B 2B 3B 4B
                           0 \times 34 - 7 \mid DB \mid 0B \mid DB \mid 0B
                           0x38-B|EC 0C EC 0C
                                                    PPN 110; valid 1
0x18-Bl1C 2C 3C 4C
                                                    M[110 \ 001 \ (0x31)] = 0x0A
0x1C-F|1C 2C 3C 4C
                           0x3C-FIFC 0C FC 0C
```

 $0 \times 131 = 1 \ 0011 \ 0001$

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused page table base register 0x20; translate virtual address 0x131

```
physical bytes
                           physical <sub>bytes</sub>
                          addresses
addresses
                                                    0x20 + 4 \times 1 = 0x24
0x00-3|00 11 22 33
                           0x20-3|00 91 72 13
                                                   PTE 1 value:
                           0x24-7D4 F5 36 07
0 \times 04 - 7 | 44 55 66 77
                                                    0 \times D4 = 1101 \ 0100
0x08-B|88 99 AA BB
                           0x28-B|89 9A AB BC
                                                    PPN 110. valid 1
0x0C-FCC DD EE FF
                           0x2C-FCD DE EF F0
                                                    PTE 2 addr:
0 \times 10 - 3 | 1A 2A 3A 4A
                           0x30-3|BA OA BA OA
                                                    110 000 + 110 \times 1 = 0x36
                                                    PTE 2 value: 0xDB
0 \times 14 - 7 | 1B 2B 3B 4B
                           0 \times 34 - 7 \mid DB \mid 0B \mid DB \mid 0B
                           0x38-B|EC 0C EC 0C
                                                    PPN 110; valid 1
0x18-Bl1C 2C 3C 4C
                                                    M[110 \ 001 \ (0x31)] = 0x0A
0x1C-F|1C 2C 3C 4C
                           0x3C-FIFC 0C FC 0C
```

 $0 \times 131 = 1 \quad 0011 \quad 0001$

2-level splitting

- 9-bit virtual address
- 6-bit physical address

- 8-byte pages \rightarrow 3-bit page offset (bottom bits)
- 9-bit VA: 6 bit VPN + 3 bit PO
- 6-bit PA: 3 bit PPN + 3 bit PO

- 8 entry page tables \rightarrow 3-bit VPN parts
- 9-bit VA: 3 bit VPN part 1; 3 bit VPN part 2

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused; page table base register 0x10; translate virtual address 0x109

physical addresses	byte	25			physical addresses	byt	es		
addresses					addresses				
0x00-3	00	11	22	33	0x20-3	D0	D1	D2	D3
0x04-7	44	55	66	77	0x24-7	D4	D5	D6	D7
0x08-B	88	99	AA	ВВ	0x28-B	89	9A	AB	ВС
0x0C-F	CC	DD	EE	FF	0x2C-F	CD	DE	EF	F0
0x10-3	1A	2A	5A	4A	0x30-3	ВА	0A	ВА	0Α
0x14-7	1B	2B	3B	4B	0x34-7	DΒ	0B	DB	0B
0x18-B	1C	2C	3C	4C	0x38-B	EC	0C	EC	0C
0x1C-F	1C	2C	3C	4C	0x3C-F	FC	0C	FC	0C

Ö

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

```
page table base register 0x10; translate virtual address 0x109
```

```
physical bytes
                            physical bytes
                                                         0 \times 109 = 100 \ 011 \ 001
```

audi esses		, (PIEI at:
0x00-3 00 11 22 33	0x20-3 D0 D1 D2 D3	$0 \times 10 + PTF$ size times 4
$0 \times 04 - 744556677$	$0 \times 24 - 7 \times 104 \times 105 \times 106 \times 107$	DTE 1. Ox1D at Ox14

0,04 1 44 33	00 11 0 24 1	04 03 00 01	PIE I: OXIB at OXI4
0x08-B88 99	AA BB 0x28-E	89 9A AB BC	PTE 1: PPN 000 (0) valid 1
0x0C-FCC DD	EE FF 0x2C-F	CD DE EF F0	(second table at:
0.10 214 24		DA OA DA OA	

0×10-3 <mark>1A 2A 5A 4A</mark>	0x30-3BA 0A BA 0A	0 (000) times page size = 0x00
0x14-7 1B 2B 3B 4B	0x34-7 DB 0B DB 0B	PTE 2: 0x33 at 0x03
0x18-B1C 2C 3C 4C	0x38-BEC 0C EC 0C	PTE 2: PPN 001 (1) valid 1
0x1C-F1C 2C 3C 4C	$0 \times 3C - FFC OC FC OC$	$0.01 0.01 = 0.09 \rightarrow 0.099$

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

```
page table base register 0x10; translate virtual address 0x109
   nhygical
```

physical bytes	bytes	$0 \times 109 = 100 \ 011 \ 001$
addresses	addresses [*]	, (PTE 1 at:
0x00-300 11 22 33	0x20-3 D0 D1 D2 D3	$0 \times 10 + PTE$ size times 4 (100))
		+ 0x10 $+$ P $+$ E Size times 4 (100))

0x00-300 11 22 33	0x20-3 D0 D1 D2 D3	0x10 + PTE size 1
$0 \times 04 - 744556677$		

0x08-B88 99 AA BB 0x28-B89 9A AB BC PTE 1: PPN 000 (0) v	
	valid 1
0x0C-FCC DD EE FF 0x2C-FCD DE EF F0 (second table at:	

		(Second table at.
0×10-3 1A 2A 5A 4A	0x30-3BA 0A BA 0A	$0 (000)$ times page size $= 0 \times 00$
0x14-7 1B 2B 3B 4B		<i>PTE 2:</i> 0x33 at 0x03
0x18-B1C 2C 3C 4C	0x38-BEC 0C EC 0C	PTE 2: PPN 001 (1) valid 1
0.10 [10.20.20.40]	0,,20 5 5 00 50 00	$0.01 \ 0.01 - 0.000 \ 0.000$

0x1C-F|1C 2C 3C 4C

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

```
page table base register 0 \times 10; translate virtual address 0 \times 109
```

physical bytes physical bytes $0 \times 109 = 100 \ 011 \ 001$ addresses (PTE 1 at:

0x00-3|00 11 22 33 0x20-3|D0 D1 D2 D3 0x10 + PTE size times 4 (100))

0x04-7|44 55 66 77 0x24-7D4 D5 D6 D7 PTF 1: 0x1B at 0x14

0x08-B|88 99 AA BB 0x28-B|89 9A AB BC PTE 1: PPN 000 (0) valid 1 0x0C-FCC DD EE FF 0x2C-FCD DE EF F0 (second table at:

0x10-3|1A 2A 5A 4A 0x30-3|BA 0A BA 0A 0 (000) times page size = 0×00) $0 \times 14 - 7 \mid 1B \mid 2B \mid 3B \mid 4B \mid$ $0 \times 34 - 7 \mid DB \mid 0B \mid DB \mid 0B$ PTF 2: 0x33 at 0x03 0x18-Bl1C 2C 3C 4C 0x38-BEC 0C EC 0C PTE 2: PPN 001 (1) valid 1 $001 \ 001 = 0x09 \rightarrow 0x99$

0x3C-FIFC 0C FC 0C

9-bit virtual addresses, 6-bit physical; 8 byte pages, 1 byte PTE

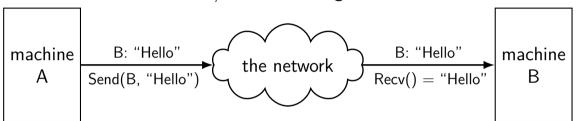
page tables 1 page; PTE: 3 bit PPN (MSB), 1 valid bit, 4 unused;

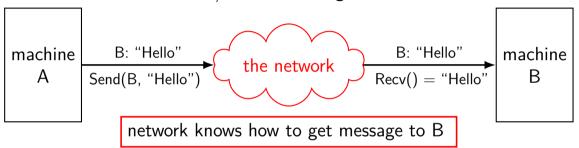
```
page table base register 0x10; translate virtual address 0x109
```

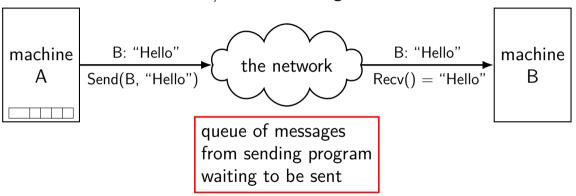
physical bytes addresses physical bytes addresses ox 109 = 100 011 001 ox 100 = 300 11 22 33 ox 20 = 300 D1 D2 D3 ox 100 = D75 ox 20 = 4 (PTE 1 at:

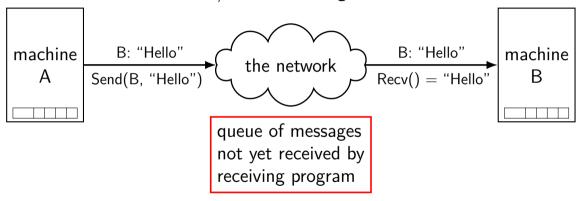
0x04-7 44 55 66 77 | 0x24-7 D4 D5 D6 D7 | PTE 1: 0x1B at 0x14 | 0x08-B 88 99 AA BB | 0x28-B 89 9A AB BC | PTE 1: PPN 000 (0) valid 1 | 0x0C-F CC DD EE FF | 0x2C-F CD DE EF F0 | (second table at:

0x10-3 1A 2A 5A 4A 0x14-7 1B 2B 3B 4B 0x18-B 1C 2C 3C 4C 0x2C-F CD DE EF F0 (second table at: 0x30-3 BA 0A BA 0A 0x30-3 BA 0A BA 0A 0x30-3 BA 0B 0B DB 0B 0x34-7 DB 0B DB 0B PTE 2: 0x33 at 0x03 0x38-B EC 0C EC 0C PTE 2: PPN 001 (1) valid 1









connections over mailboxes

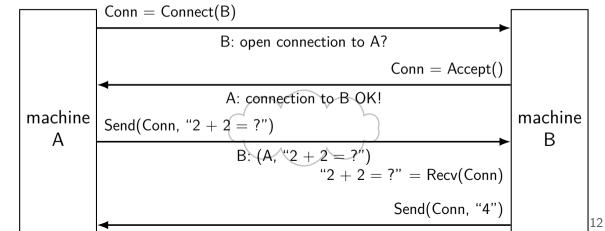
real Internet: mailbox-style communication send packets to particular mailboxes no gaurentee on order, when received

sockets implemented on top of this

conections

connections: two-way channel for messages

extra operations: connect, accept



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 mea	nings
transport	TCP, UDP,	reach correct prog	gram,
		reliablity/streams	
network	IPv4, IPv6,	reach correct ma	chine
		(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,	
		reliablity/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

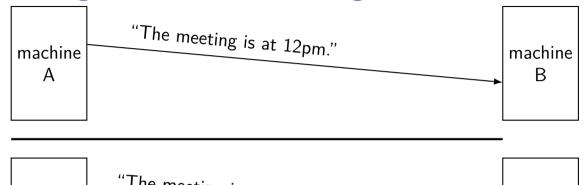
messages lost

messages delayed/reordered

messages limited in size

messages corrupted

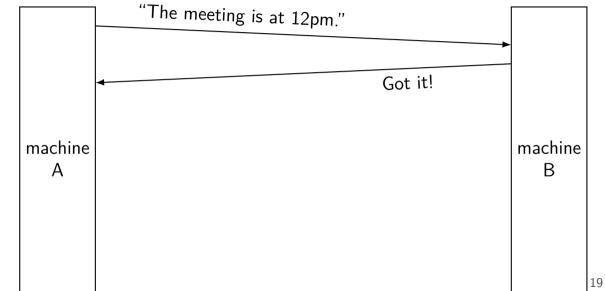
dealing with network message lost



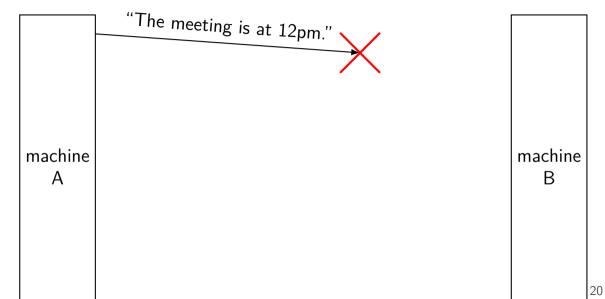
machine A "The meeting is at 12pm."

machine B

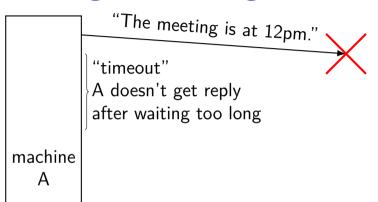
handling lost message: acknowledgements



handling lost message

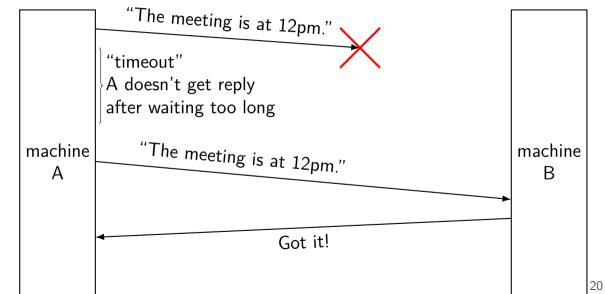


handling lost message

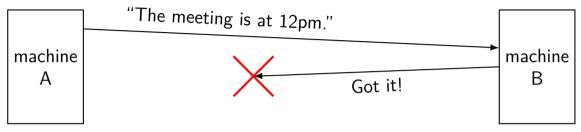


machine B

handling lost message



exercise: lost acknowledgement



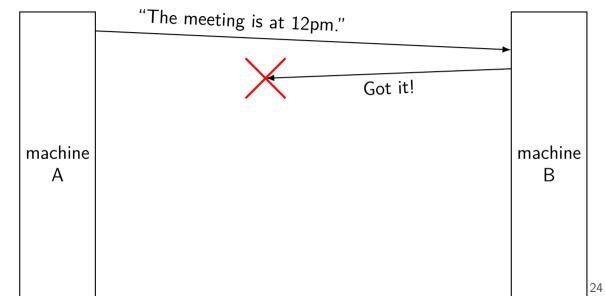
exercise: how to fix this?

- A. machine A needs to send "Got 'got it!"
- B. machine B should resend "Got it!" on its own
- C. machine A should resend the original message on its own
- D. none of these

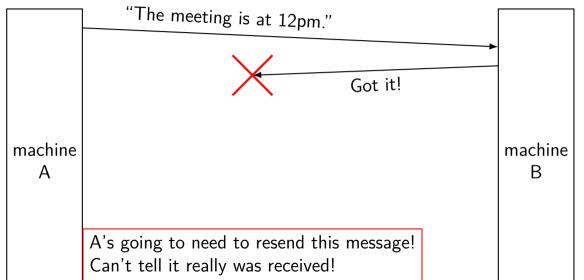
answers

```
send "Got 'got it!' "?
     same problem: Now send 'Got Got Got it'?
resend "Got it!" own its own?
     how many times? — B doesn't have that info
resend original message?
     ves!
     as far as machine A can be, exact same situation as losing original
     message
```

lost acknowledgements

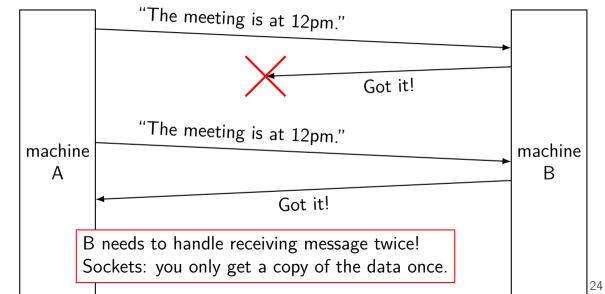


lost acknowledgements



24

lost acknowledgements



network limitations/failures

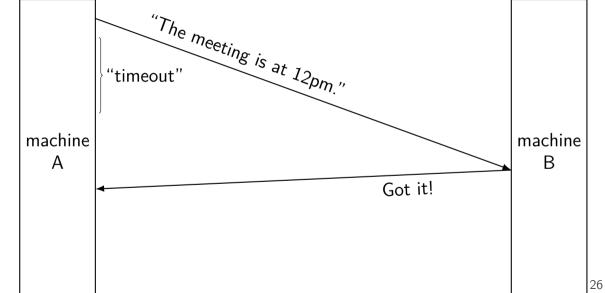
messages lost

messages delayed/reordered

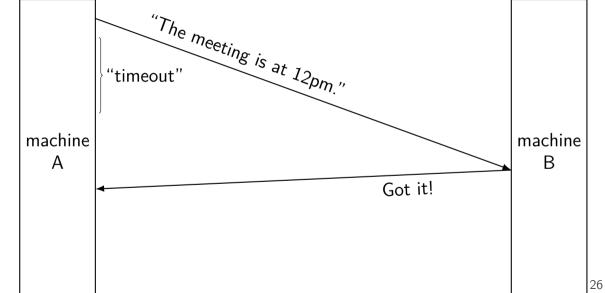
messages limited in size

messages corrupted

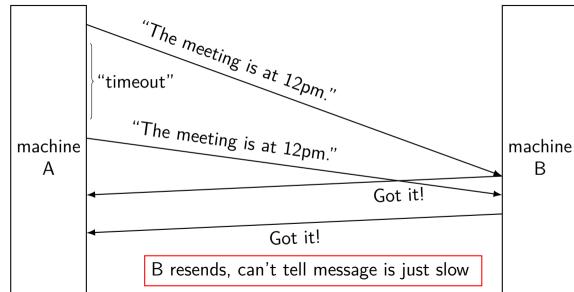
delayed message



delayed message

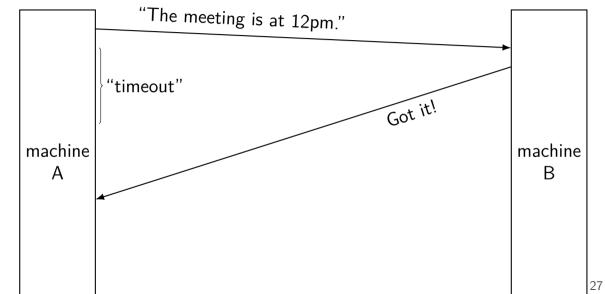


delayed message

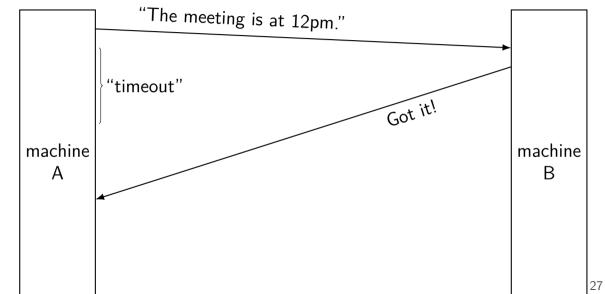


26

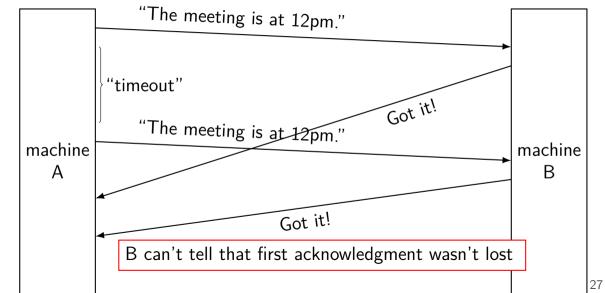
delayed acknowledgements



delayed acknowledgements



delayed acknowledgements



network limitations/failures

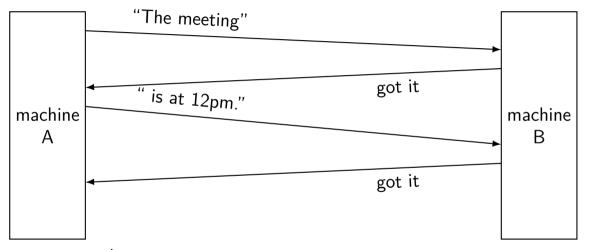
messages lost

messages delayed/reordered

messages limited in size

messages corrupted

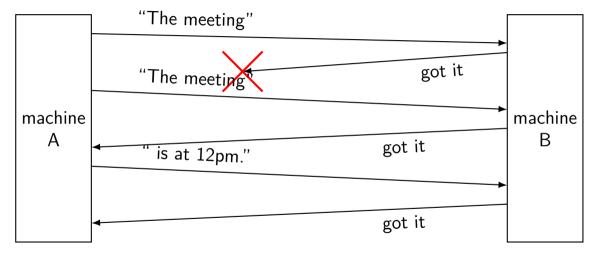
splitting messages: try 1



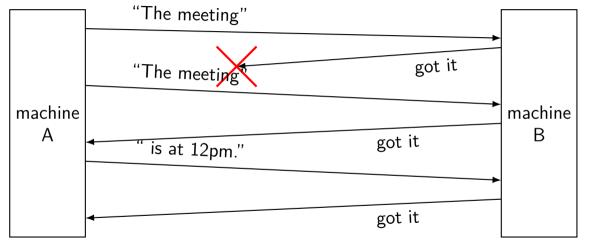
reconstructed message:

The meeting is at 12pm.

splitting messages: try 1 — problem 1



splitting messages: try 1 — problem 1



reconstructed message:

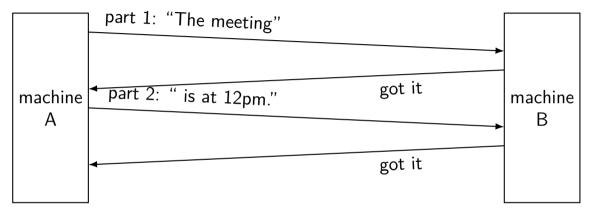
The meeting The meeting is at 12pm.

exercise: other problems?

other scenarios where we'd also have problems?

- 1. message (instead of acknowledgment) is lost
- 2. first message from machine A is delayed a long time by network
- 3. acknowledgment of second message lost instead of first

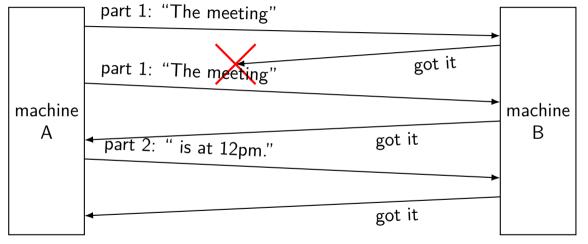
splitting messages: try 2



reconstructed message:

The meeting is at 12pm.

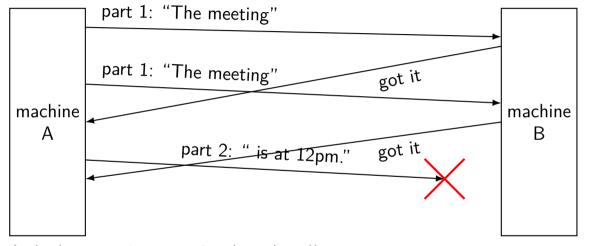
splitting messages: try 2 — missed ack



reconstructed message:

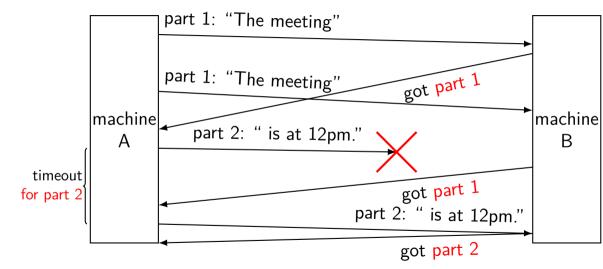
The meeting is at 12pm.

splitting messages: try 2 — problem



A thinks: part 1 + part 2 acknowleged!

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 Hash("message") = 0xABCDEF12
then send "0xABCDEF12,message"
```

when receiving, recompute hash pretend message lost if does not match

"checksum"

these hashes commonly called "checksums"

in UDP/TCP, hash function: treat bytes of messages as array of integers; then add integers together

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

backup slides