changelog

17 Feb 2023: memory access example: shift things so address labels are not off from numbers shown

last time (1)

assignment Q&A

multi-level page table lookup

unreliable networks

"best effort" model of the internet

- limited message size
- sometimes messages lost
- sometimes messages delayed/reordered
- sometimes messages corrupted

last time (2)

sequence numbers

acknowledgments

- someone needs to resend after timeout
- can get lost, that's okay

checksums

some themes in anonymous feedback

pagetable difficulty

lab difficulty

quizzes: how many/etc.

Quiz Q3

1 first-level page table with

- a valid entry pointing to a second-level page table with 512 valid entries
- a valid entry pointing to a second-level page table with (1000-512) valid entries and a few invalid entries and 510 invalid entries

three 4096-byte page tables

Quiz Q4

0x120008 = PTBR + VPN part 1 × PTE size =
0x1200000 + VPN part 1 × 8 → VPN part 1 = 1

0x123040 =
PPN from 1st level × page size + VPN part 2 × PTE size →
VPN part 2 = 8

0x6010 =
PPN from 2nd level × page size + page offset → page offset = 0x10

Quiz Q5

"It then runs a function, whose machine code is loaded at addresses 0x2040-0x2072, which writes 3 8-byte values to the stack at addresses 0xFFF8, 0xFFF0, and 0xFFE8."

page at 0x2000–0x2FFF

- code loaded on first instruction's page fault
- can't tell processor about only part of page being loaded

page at 0xF000–0xFFFF

- whole page of stack allocated on first access

HW difficulty

"...I feel like several components of the assignment we have not fully learned and some we just learned about in lecture today. Additionally, I think while a checkpoint is a reasonable idea, we could all benefit from the extra time and just have the first two parts be due next week. I have been in office hours the last two days and it seems like barely any students know what is going on."

"While the quiz made sense and was related to the lectures and readings, this homework assignment has a lot of things that you need to rely on TA's or word of mouth for. For example, how would we know that we need to memset after posix_memalign if we don't even know how to look that up..."

"I feel like the content of the lectures is too far removed from what we are asked to do in the homeworks...."

mistakes I made with homework (1)

overestimated C familiarity from CSO1

- a lot of problems from C pointer issues
- fails in ways that are not intuitive, especially if you aren't checking every step
- why I assumed understanding manpage for posix_memalign was not big deal
- future: warmup assignment should probably review C pointer stuff somehow
- b/c of this, put halfway point of assignment at wrong place

in future semesters, need to plan more lecture time for virtual memory

mistakes I made with homework (2)

some things in writeup are/were too easy to miss

- page table entry format
- physical page number v physical address
- what things need to be allocated

need more structure re: testing

- students just using code in assignment + autograder was not the intention
- seems like (based on submissions) many students writing a lot of code before testing it, rather than testing in small pieces

pagetable grading

submission 1 (25% of normal homework)

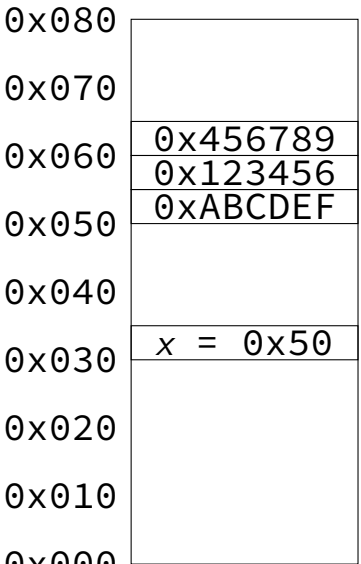
- 32% LEVELS = 1 reasonable attempt
- 64% reasonable attempt on two other items
- 3% code style

submission 2 (25% of normal homework)

- 50% everything present
- 40% LEVELS = 1 functionality
- 10% LEVELS > 1 functionality

submission 3 (200% of normal homework)

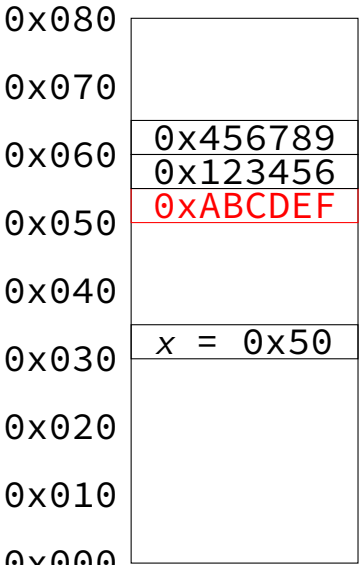
some pointer stuff




```
size_t x = 0x50;
```

~~`*x`~~ (compile-time error)

some pointer stuff



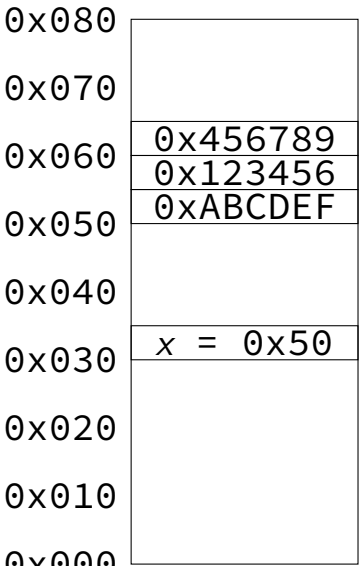

```
size_t x = 0x50;
```

~~`*x`~~ (compile-time error)

```
size_t *ptr;
ptr = (size_t *) x;
*ptr == 0xABCDEF
```

```
*((size_t *) x) == 0xABCDEF
```

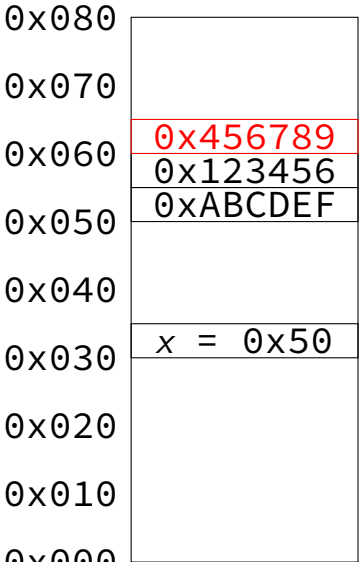
some pointer stuff




```
size_t x = 0x50;
```

~~`x[2]`~~ (compile-time error)

some pointer stuff



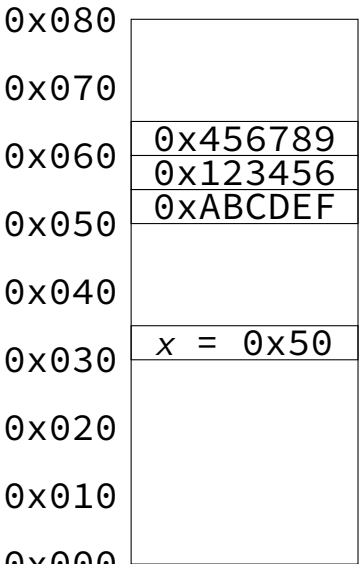

```
size_t x = 0x50;
```

~~`x[2]`~~ (compile-time error)

```
size_t addr = x + 16;
size_t *ptr;
ptr = (size_t *) addr;
*ptr == 0x456789
```

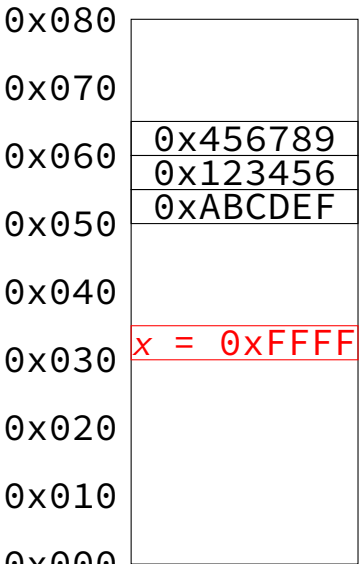
```
size_t *ptr;
ptr = (size_t *) x;
ptr[2] == 0x456789
```

some pointer stuff



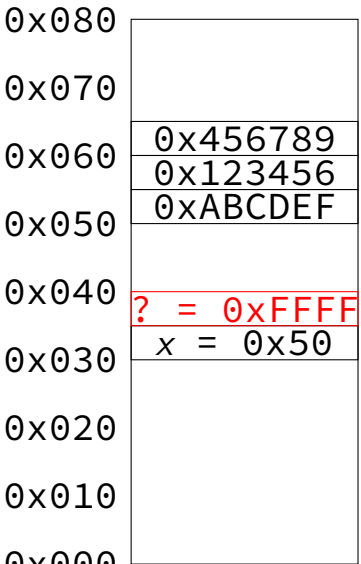

```
size_t x = 0x50;

void change_arg(size_t *arg) {
     *arg = 0xFFFF;
}
```

some pointer stuff



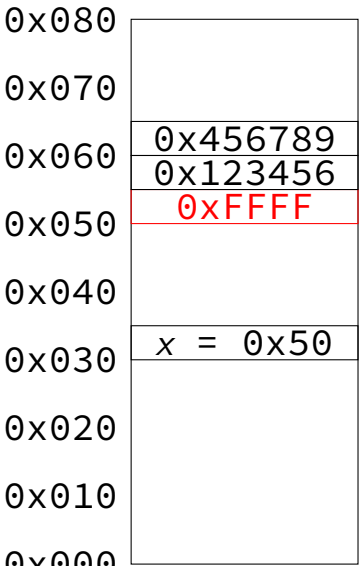

```
size_t x = 0x50;

void change_arg(size_t *arg) {
     *arg = 0xFFFF;
}

change_arg(&x);
change_arg((size_t*) 0x30);
```

some pointer stuff



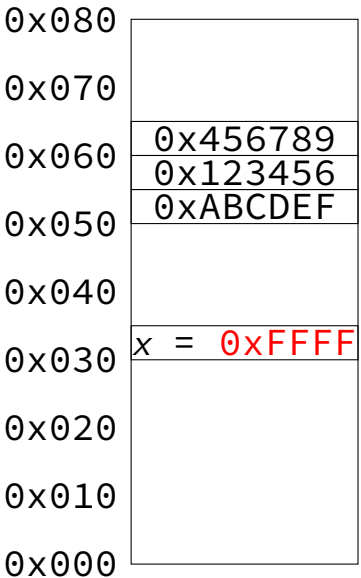

```
size_t x = 0x50;

void change_arg(size_t *arg) {
    *arg = 0xFFFF;
}

change_arg(&x + 1);
change_arg((size_t*) 0x38);
```

some pointer stuff



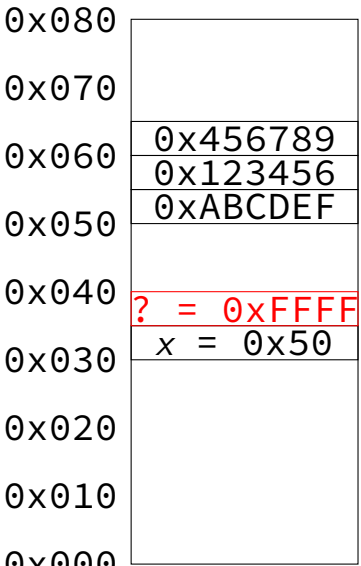

```
size_t x = 0x50;

void change_arg(size_t *arg) {
    *arg = 0xFFFF;
}

change_arg((size_t *) x);
change_arg((size_t *) 0x50);
```

some pointer stuff



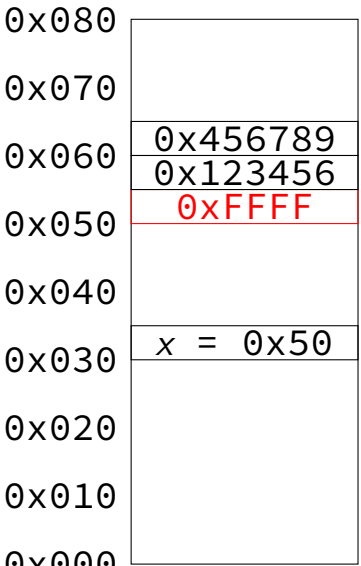

```
void *x = (void *) 0x50

void change_arg(void **arg) {
     *arg = (void *) 0xFFFF;
}

change_arg((void **) &x);
change_arg((void **) 0x30);
```

some pointer stuff




```
void *x = (void *) 0x50

void change_arg(void **arg) {
     *arg = (void *) 0xFFFF;
}

change_arg(&x + 1);
change_arg((void **) 0x38);
```

some pointer stuff




```
void *x = (void *) 0x50

void change_arg(void **arg) {
     *arg = (void *) 0xFFFF;
}

change_arg((void **) x);
change_arg((void **) 0x50);
```

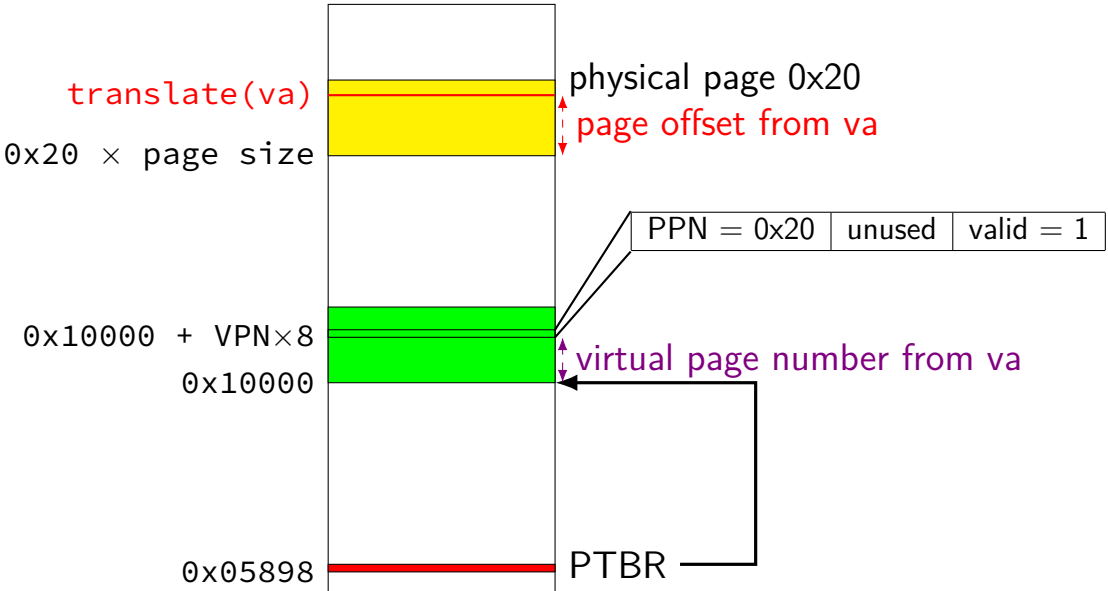
address/page table entry format

(with POBITS=12, LEVELS=1)

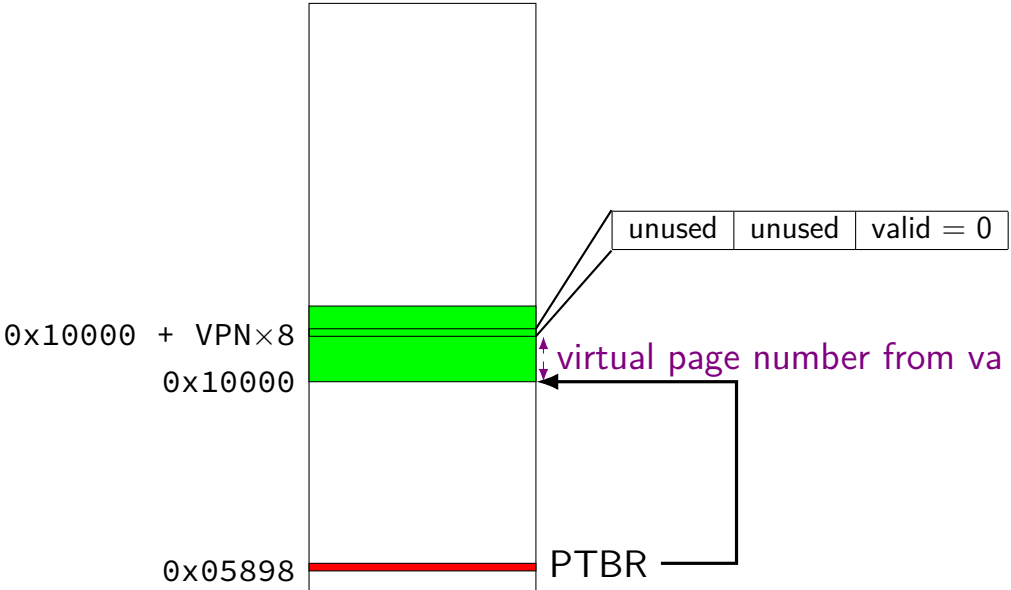
	bits 63–21	bits 20–12	bits 11–1	bit 0
page table entry	physical page number		unused	valid bit
virtual address	unused	virtual page number	page offset	
physical address	physical page number		page offset	

in assignment: value from posix_memalign = physical address

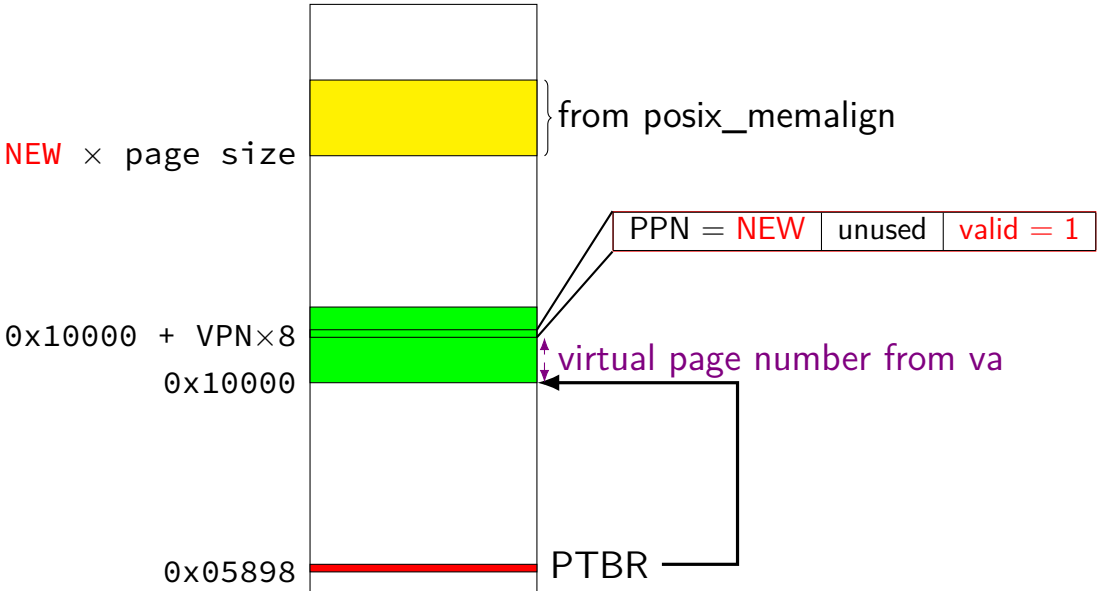
pa = translate(va)



page_allocate(va)



page_allocate(va)



next week's lab

code review your submission 2 with other students

must be in person!

can't attend lab? talk to me!

use the feedback to improve your submission 3

lab difficulty

"I wish we could at least get more explanation for what is going on in the networking lab. I understood Tuesday's lecture enough to at least get the concept, but the lab write-up itself was pretty opaque and it felt like we were being thrown into the deep end to actually implement the networking. I spent the whole 75 minutes in lab just going over the reading and trying to figure out what exactly we were supposed to do..."

lab difficulty

was surprised by confusion re: recvd() function + setTimeout()

- oops! should have realized you haven't seen these kinds of interfaces before

probably need an introduction to this type of interface in lecture in the future

callback-based programming (1)

```
/* library code you don't write */
void mainLoop() {
    while (true) {
        Event event = getNextEvent();
        if (event.type == RECIEVED) {
            recvd(...);
        } else if (event.type == TIMEOUT) {
            (event.timeout_function)(...);
        }
        ...
    }
}
```

callback-based programming (2)

```
/* your code, called by library */
void recvd(...) {
    ...
    setTimeout(..., timerCallback, ...);
}

void timerCalback(...) {
    ...
}
```

callback-based programming

writing scripts in a webpage

many graphical user interface libraries

sometimes servers that handle lots of connections

protocol

GET0 — start

other end acknowledges by giving data
if they don't acknowledge, you need to send again

ACKn

request message n + 1 by acknowledging message n
not quite same purpose as acknowledgments in lecture examples
(in lab, the respoonse is your 'acknowledgment' of your request;
you retry if you don't get it)

feedback re: quizzes

"I would appreciate if the quizzes were a little longer. We learn a lot in this class and I don't think 5 questions (sometimes with no partial credit) is the best representation of our skills."

"would you ever consider dropping the lowest quiz grade?"

"I've found that the quizzes are incredibly difficult...I feel that the scope of the quizzes is way beyond the lecture material and readings...Maybe lecture material and/or readings could more closely align with the quiz questions, so that we are better prepared."

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

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

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::2005`
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:da`
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, reliablity/streams
network	IPv4, IPv6, ...	reach correct machine (across networks)
link	Ethernet, Wi-Fi, ...	coordinate shared wire/radio
physical	...	encode bits for wire/radio

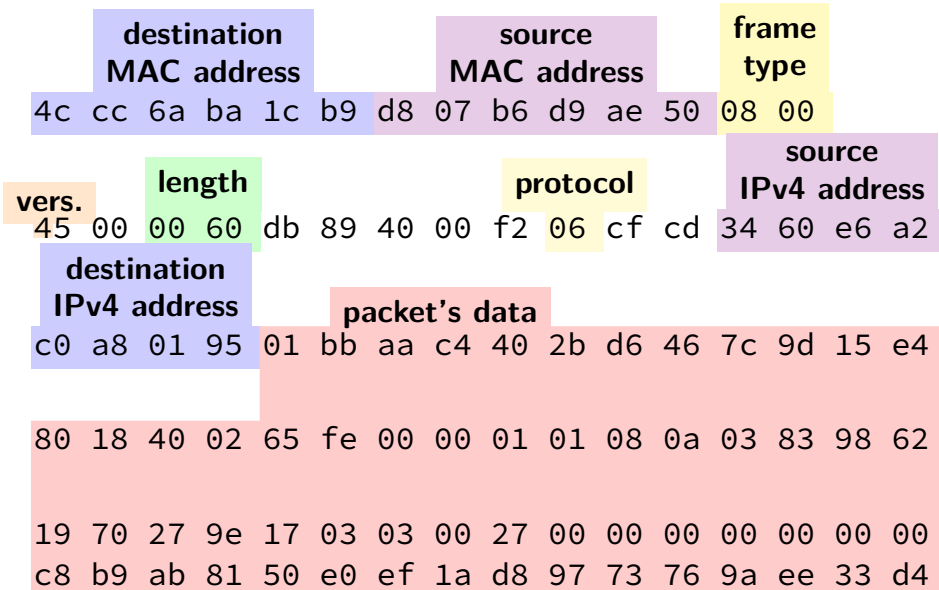
an Ethernet frame

destination MAC address	source MAC address	frame type
4c cc 6a ba 1c b9	d8 07 b6 d9 ae 50	08 00

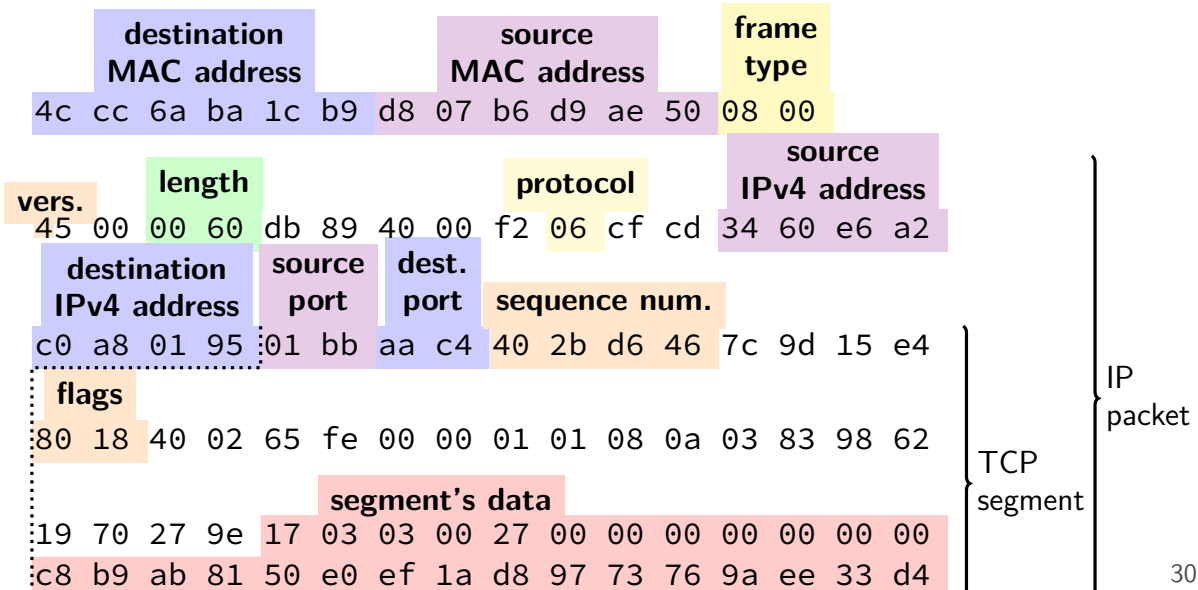
frame's data

```
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
```

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*)

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?

link layer reliablity?

Ethernet + Wifi have checksums

Q1: Why doesn't this give us uncorrupted messages?
Why do we still have checksums at the higher layers?

Q2: What's a benefit of doing this if we're also doing it in the higher layer?

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

the network layer

the Internet Protocool (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

32-bit numbers

typically written like 128.143.67.11

- four 8-bit decimal values separated by dots
- first part is most significant
- same as $128 \cdot 256^3 + 143 \cdot 256^2 + 67 \cdot 256 + 11 = 2\,156\,782\,459$

organizations get blocks of IPs

- e.g. UVa has 128.143.0.0–128.143.255.255
- e.g. Google has 216.58.192.0–216.58.223.255 and 74.125.0.0–74.125.255.255 and 35.192.0.0–35.207.255.255

some IPs reserved for non-Internet use (127.*, 10.*, 192.168.*)

IPv6 addresses

IPv6 like IPv4, but with 128-bit numbers

written in hex, 16-bit parts, seperated by colons (`:`)

strings of 0s represented by double-colons (`::`)

typically given to users in blocks of 2^{80} or 2^{64} addresses

- no need for address translation?

`2607:f8b0:400d:c00::6a` =
`2607:f8b0:400d:0c00:0000:0000:0000:006a`

- `2607f8b0400d0c0000000000000006a`$_{\text{SIXTEEN}}$

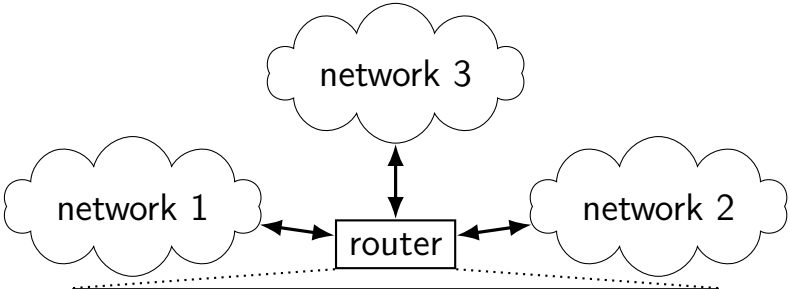
selected special IPv6 addresses

`::1` = localhost

anything starting with `fe80` = link-local addresses

- never forwarded by routers

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

selected special IPv4 addresses

127.0.0.0 — 127.255.255.255 — localhost

- AKA loopback
- the machine we're on
- typically only 127.0.0.1 is used

192.168.0.0–192.168.255.255 and
10.0.0.0–10.255.255.255 and
172.16.0.0–172.31.255.255

- "private" IP addresses
- not used on the Internet
- commonly connected to Internet with network address translation
- also 100.64.0.0–100.127.255.255 (but with restrictions)

169.254.0.0-169.254.255.255

- link-local addresses — 'never' forwarded by routers