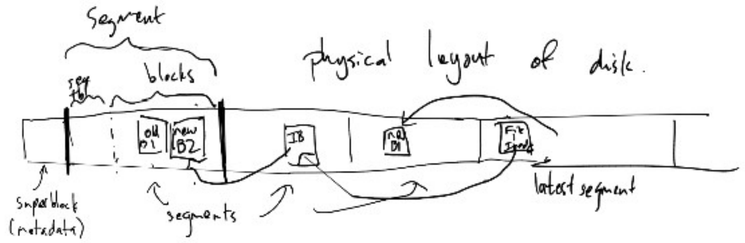
Lecture 20: finish LFS, start networking

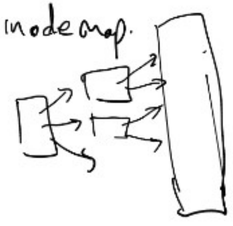
- LFS
  - Compaction
  - benefits

- Networking
  - Ethernet
  - Layers



physical layout of disk.

- only write into last segment on disk (tail of a log).
- to overwrite/update file: create new copy in curr. segment
  - update pointers leading to block.
  - root of FS always in curr. seg.



Compacting a segment:

- "rewrite" each block in segment (as if we're updating the file).
- store "segment table" in each segment indicating identity of each block in segment (eg. first block in segment is DB 37 of F1. -or- IB#5 contained in DIB#3 in file 8

- compactor compares log. identities of blocks in segment to current addresses (in current segment): if still latest ver. log. make new copy (make stale) after running: all blocks stale, reuse seg.

in UFS: directory contains disk addresses of nodes of files in directory.

in LFS: directories contain inode #s, need "inode map" mapping log. inode #s to phys. disk addresses

---

might not compact all segments: might only compact if segment < 75% in use.

- need list of free segments.

Few bonus advantages of LFS:

- single file can be spread all over disk (assumed reads cached, don't worry).
  - actually all blocks written at same time in same segment.
    usually read in same order as write.
- LFS great for SSDs.
  - write to SSD (flash): (~1MB)
    - clear (or flash) an entire large block of addresses.
    - can write over cleared space.
    - like to do lots of writes at once.
      - like to not have to do partial writes.
  - SSD degrades after many writes.
    - ideal: uniformly distribute writes over disk (throughout disk lifetime)
      "write balancing"

- Append-only data structure useful elsewhere:
  - Bitcoin (blockchain)
    - uses crypto to enforce append-only nature.
  - Distributed systems.
    hard to maintain consistency.

Networking

- Want to connect multiple computers.
- Network "stack"

OSI layers

- Physical layer: connecting computers to each other (directly).
  - wired connections, wireless connections
  - protocol: voltages, bit-level encodings, frequency
- Link layer: manages who can send/receive at any given time
  - Ethernet.
- Internetworking layer: IP (internet protocol)
  - how do I forward data from one network to the other
- Transport layer: (UDP: simple packet-oriented protocol, instead of sending individual small messages, send stream of data. TCP: transmission control protocol: stream of communication, handles overload of network)
- Session layer: opening / closing connections
  - Software layer: sockets.
- Presentation layer: layout data structures as bits/bytes.
- Application layer: what you're sending/receiving as application, how to respond to requests.
  - eg. HTTP: how to request web pages.
  - e.g. FTP: how to send files back & forth.

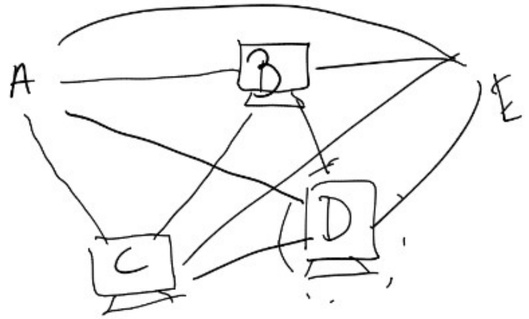
HW

OS

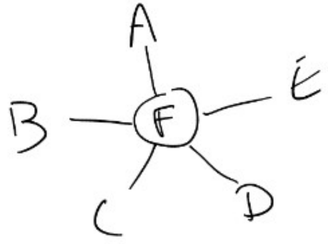
Application

How to connect small # of computers (e.g. 3-5)
(local area network / LAN)

(clique)



wires/connection:
$O(n^2)$



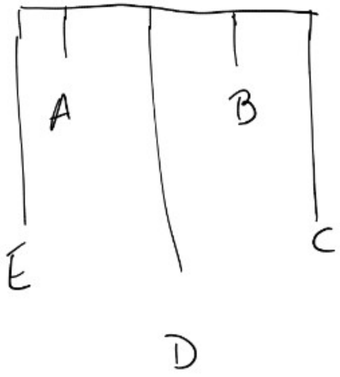
(Star topology)

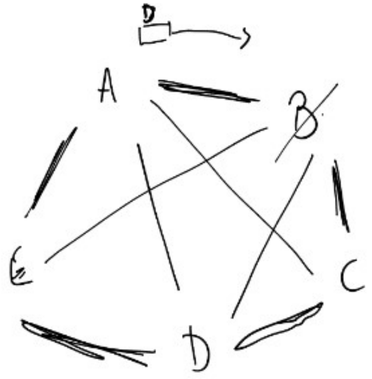
wires (connections)/host:
$O(n)$

each 'host': 1 connection (except F)

centralized - F does a lot (needs complicated device)

- F is a single point of failure.





Token ring: Complicated