# CS162
# Operating Systems and Systems Programming
# Lecture 20

## Distributed Systems

November 9, 2005

Prof. John Kubiatowicz

http://inst.eecs.berkeley.edu/~cs162

## Review: How do we actually access files?

- All information about a file contained in its file header
  - UNIX calls this an "inode"
    - » Inodes are global resources identified by index ("inumber")
  - Once you load the header structure, all the other blocks of the file are locatable
- Naming: The process by which a system translates from user-visible names to system resources
  - In the case of files, need to translate from strings (textual names) or icons to inumbers/inodes
- Name Resolution: The process of converting a logical name into a physical resource (like a file)
  - Traverse succession of directories until reach target file
  - Global file system: May be spread across the network
- Directory: a relation used for naming
  - Just a table of (file name, inumber) pairs
  - Directories often stored in files
    - » Reuse of existing mechanism
    - » Directory named by inode/inumber like other files



## Goals for Today

- File Caching
- Data Durability
- Distributed Systems

Note: Some slides and/or pictures in the following are adapted from slides ©2005 Silberschatz, Galvin, and Gagne



## File System Caching

- Key Idea: Exploit locality by caching data in memory
  - Name translations: Mapping from paths→inodes
  - Disk blocks: Mapping from block address→disk content
- Buffer Cache: Memory used to cache kernel resources, including disk blocks and name translations
  - Can contain "dirty" blocks (blocks yet on disk)
- Replacement policy? LRU
  - Can afford overhead of timestamps for each disk block
  - Advantages:
    - » Works very well for name translation
    - » Works well in general as long as memory is big enough to accommodate a host's working set of files.
  - Disadvantages:
    - » Fails when some application scans through file system, thereby flushing the cache with data used only once
    - » Example: `find . -exec grep foo {} \;`
- Other Replacement Policies?
  - Some systems allow applications to request other policies
  - Example, 'Use Once':
    - » File system can discard blocks as soon as they are used

## File System Caching (con't)

- Cache Size: How much memory should the OS allocate to the buffer cache vs virtual memory?
  - Too much memory to the file system cache ⇒ won't be able to run many applications at once
  - Too little memory to file system cache ⇒ many applications may run slowly (disk caching not effective)
  - Solution: adjust boundary dynamically so that the disk access rates for paging and file access are balanced
- **Read Ahead Prefetching:** fetch sequential blocks early
  - Key Idea: exploit fact that most common file access is sequential by prefetching subsequent disk blocks ahead of current read request (if they are not already in memory)
  - Elevator algorithm can efficiently interleave groups of prefetches from concurrent applications
  - How much to prefetch?
    - » Too many imposes delays on requests by other applications
    - » Too few causes many seeks (and rotational delays) among concurrent file requests



## File System Caching (con't)

- **Delayed Writes:** Writes to files not immediately sent out to disk
  - Instead, `write()` copies data from user space buffer to kernel buffer (in cache)
    - » Enabled by presence of buffer cache: can leave written file blocks in cache for a while
    - » If some other application tries to read data before written to disk, file system will read from cache
  - Flushed to disk periodically (e.g. in UNIX, every 30 sec)
  - Advantages:
    - » Disk scheduler can efficiently order lots of requests
    - » Disk allocation algorithm can be run with correct size value for a file
    - » Some files need never get written to disk! (e..g temporary scratch files written /tmp often don't exist for 30 sec)
  - Disadvantages
    - » What if system crashes before file has been written out?
    - » Worse yet, what if system crashes before a directory file has been written out? (lose pointer to inode!)



## Important "ilities"

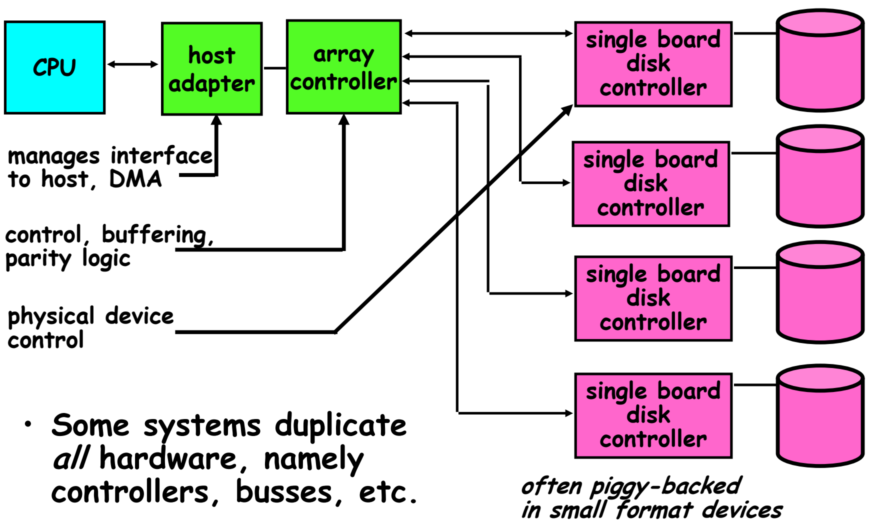
- **Availability:** the probability that the system can accept and process requests
  - Often measured in "nines" of probability. So, a 99.9% probability is considered "3-nines of availability"
  - Key idea here is independence of failures
- **Durability:** the ability of a system to recover data despite faults
  - This idea is fault tolerance applied to data
  - Doesn't necessarily imply availability: information on pyramids was very durable, but could not be accessed until discovery of Rosetta Stone
- **Reliability:** the ability of a system or component to perform its required functions under stated conditions for a specified period of time (IEEE definition)
  - Usually stronger than simply availability: means that the system is not only "up", but also working correctly
  - Includes availability, security, fault tolerance/durability
  - Must make sure data survives system crashes, disk crashes, other problems



## How to make file system durable?

- Disk blocks contain Reed-Solomon error correcting codes (ECC) to deal with small defects in disk drive
  - Can allow recovery of data from small media defects
- Make sure writes survive in short term
  - Either abandon delayed writes or
  - use special, battery-backed RAM (called non-volatile RAM or **NVRAM**) for dirty blocks in buffer cache.
- Make sure that data survives in long term
  - Need to replicate! More than one copy of data!
  - Important element: **independence of failure**
    - » Could put copies on one disk, but if disk head fails…
    - » Could put copies on different disks, but if server fails…
    - » Could put copies on different servers, but if building is struck by lightning….
    - » Could put copies on servers in different continents…
- **RAID:** Redundant Arrays of Inexpensive Disks
  - Data stored on multiple disks (redundancy)
  - Either in software or hardware
    - » In hardware case, done by disk controller; file system may not even know that there is more than one disk in use

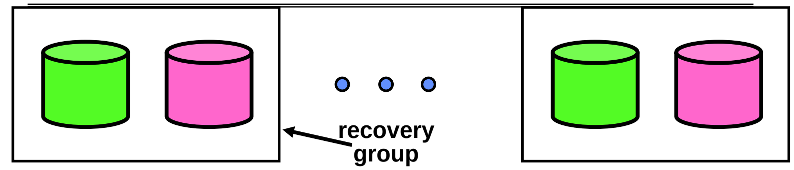
## Hardware RAID: Subsystem Organization




- Some systems duplicate *all* hardware, namely controllers, busses, etc.



## RAID 1: Disk Mirroring/Shadowing

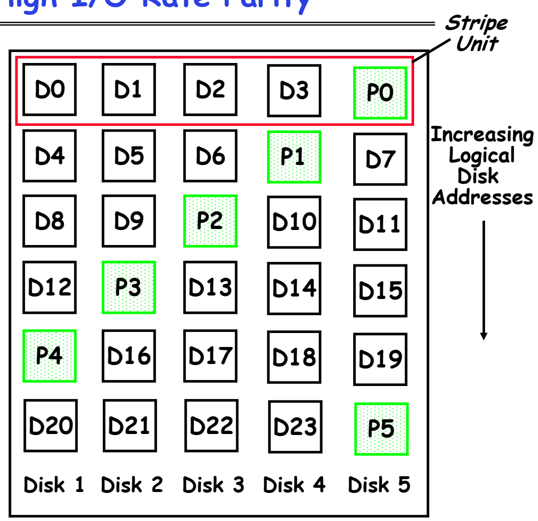



- Each disk is fully duplicated onto its "shadow"
  - For high I/O rate, high availability environments
  - Most expensive solution: 100% capacity overhead
- Bandwidth sacrificed on write:
  - Logical write = two physical writes
  - Highest bandwidth when disk heads and rotation fully synchronized (hard to do exactly)
- Reads may be optimized
  - Can have two independent reads to same data
- Recovery:
  - Disk failure ⇒ replace disk and copy data to new disk
  - Hot Spare: idle disk already attached to system to be used for immediate replacement



## RAID 5+: High I/O Rate Parity

- Data stripped across multiple disks
  - Successive blocks stored on successive (non-parity) disks
  - Increased bandwidth over single disk
- Parity block (in green) constructed by XORing data bocks in stripe
  - P0=D0⊕D1⊕D2⊕D3
  - Can destroy any one disk and still reconstruct data
  - Suppose D3 fails, then can reconstruct: D3=D0⊕D1⊕D2⊕P0



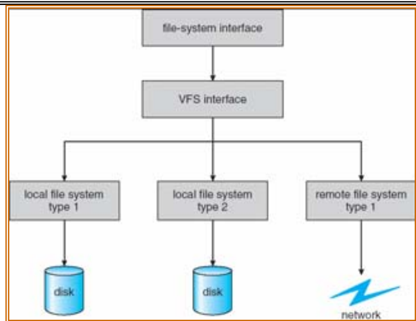

- Later in term: talk about spreading information widely across internet for durability.



## Administrivia

- My office hours
  - New office hour: Thursday 2:30-3:30
- MIDTERM II: Wednesday November 30th
  - 5:30-8:30pm, 10 Evans
  - All material from last midterm and up to Monday 11/28
  - Includes virtual memory
- Final Exam
  - December 17th, 12:30 – 3:30, 220 Hearst Gym

## Remote File Systems: Virtual File System (VFS)




- VFS: Virtual abstraction similar to local file system
  - Instead of "inodes" has "vnodes"
  - Compatible with a variety of local and remote file systems
    - » provides object-oriented way of implementing file systems
- VFS allows the same system call interface (the API) to be used for different types of file systems
  - The API is to the VFS interface, rather than any specific type of file system

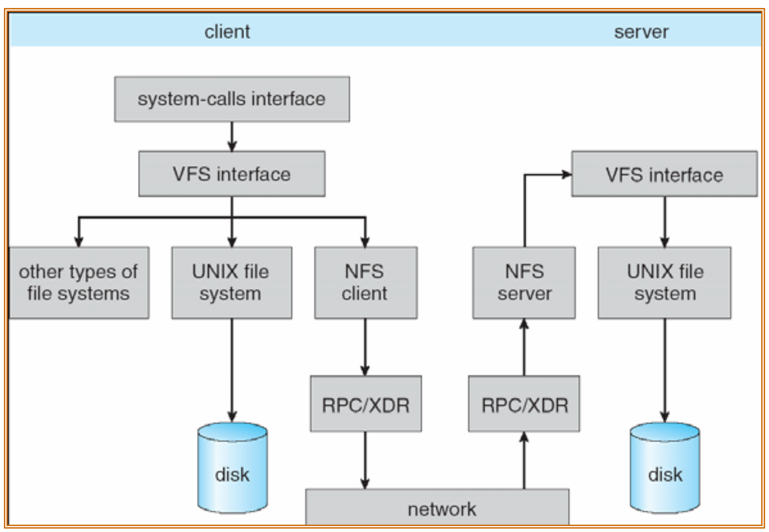


## Network File System (NFS)

- Three Layers for NFS system
  - UNIX file-system interface: open, read, write, close calls + file descriptors
  - VFS layer: distinguishes local from remote files
    - » Calls the NFS protocol procedures for remote requests
  - NFS service layer: bottom layer of the architecture
    - » Implements the NFS protocol
- NFS Protocol: remote procedure calls (RPC) for file operations on server
  - Reading/searching a directory
  - manipulating links and directories
  - accessing file attributes/reading and writing files
- NFS servers are stateless; each request provides all arguments require for execution
- Modified data must be committed to the server's disk before results are returned to the client
  - lose some of the advantages of caching
  - Can lead to weird results: write file on one client, read on other, get old data

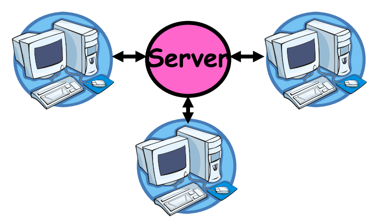


## Schematic View of NFS Architecture

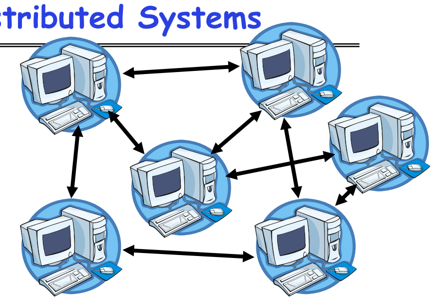





## Centralized vs Distributed Systems




Client/Server Model



Peer-to-Peer Model

- Centralized System: System in which major functions are performed by a single physical computer
  - Originally, everything on single computer
  - Later: client/server model
- Distributed System: physically separate computers working together on some task
  - Early model: multiple servers working together
    - » Probably in the same room or building
    - » Often called a "cluster"
  - Later models: peer-to-peer/wide-spread collaboration

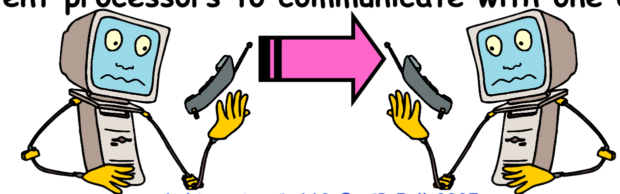
## Distributed Systems: Motivation/Issues

- Why do we want distributed systems?
  - Cheaper and easier to build lots of simple computers
  - Easier to add power incrementally
  - Users can have complete control over some components
  - Collaboration: Much easier for users to collaborate through network resources (such as network file systems)
- The *promise* of distributed systems:
  - Higher availability: one machine goes down, use another
  - Better durability: store data in multiple locations
  - More security: each piece easier to make secure
- Reality has been disappointing
  - Worse availability: depend on every machine being up
    - » Lamport: "a distributed system is one where I can't do work because some machine I've never heard of isn't working!"
  - Worse reliability: can lose data if any machine crashes
  - Worse security: anyone in world can break into system
- Coordination is more difficult
  - Must coordinate multiple copies of shared state information (using only a network)
  - What would be easy in a centralized system becomes a lot more difficult.



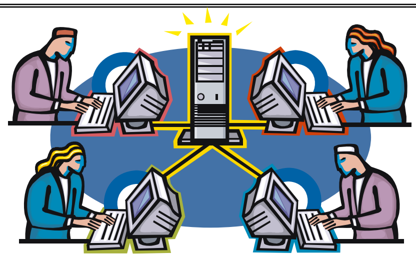
## Distributed Systems: Goals/Requirements

- Transparency: the ability of the system to mask its complexity behind a simple interface
- Possible transparencies:
  - Location: Can't tell where resources are located
  - Migration: Resources may move without the user knowing
  - Replication: Can't tell how many copies of resource exist
  - Concurrency: Can't tell how many users there are
  - Parallelism: System may speed up large jobs by spliting them into smaller pieces
  - Fault Tolerance: System may hide varoius things that go wrong in the system
- Transparency and collaboration require some way for different processors to communicate with one another





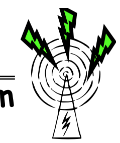
## Networking Definitions



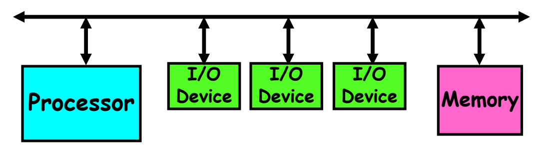
- Network: physical connection that allows two computers to communicate
- Packet: unit of transfer, sequence of bits carried over the network
  - Network carries packets from on CPU to another
  - Destination gets interrupt when packet arrives
- Protocol: agreement between two parties as to how information is to be transmitted



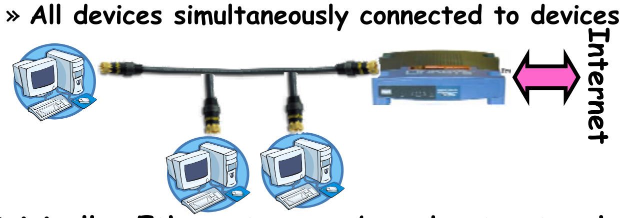
## Broadcast Networks



- Broadcast Network: Shared Communication Medium

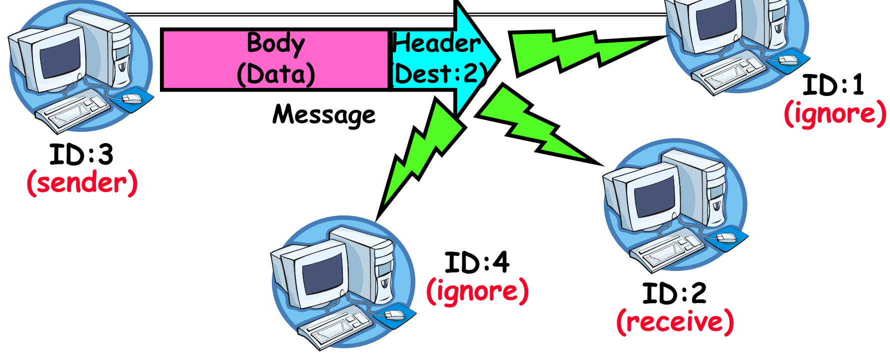



  - Shared Medium can be a set of wires
    - » Inside a computer, this is called a bus
    - » All devices simultaneously connected to devices




  - Originally, Ethernet was a broadcast network
    - » All computers on local subnet connected to one another
  - More examples (wireless: medium is air): cellular phones, GSM GPRS, EDGE, CDMA 1xRTT, and 1evDO

## Broadcast Networks Details

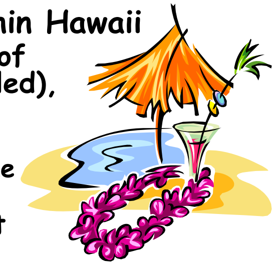



- **Delivery:** When you broadcast a packet, how does a receiver know who it is for? (packet goes to everyone!)
  - Put header on front of packet: [ Destination | Packet ]
  - Everyone gets packet, discards if not the target
  - In Ethernet, this check is done in hardware
    - » No OS interrupt if not for particular destination
  - This is layering: we're going to build complex network protocols by layering on top of the packet



## Broadcast Network Arbitration

- **Arbitration:** Act of negotiating use of shared medium
  - What if two senders try to broadcast at same time?
  - Concurrent activity but can't use shared memory to coordinate!
- Aloha network (70's): packet radio within Hawaii
  - Blind broadcast, with checksum at end of packet. If received correctly (not garbled), send back an acknowledgement. If not received correctly, discard.
    - » Need checksum anyway – in case airplane flies overhead
  - Sender waits for a while, and if doesn't get an acknowledgement, re-transmits.
  - If two senders try to send at same time, both get garbled, both simply re-send later.
  - Problem: Stability: what if load increases?
    - » More collisions ⇒ less gets through ⇒more resent ⇒ more load… ⇒ More collisions…
    - » Unfortunately: some sender may have started in clear, get scrambled without finishing



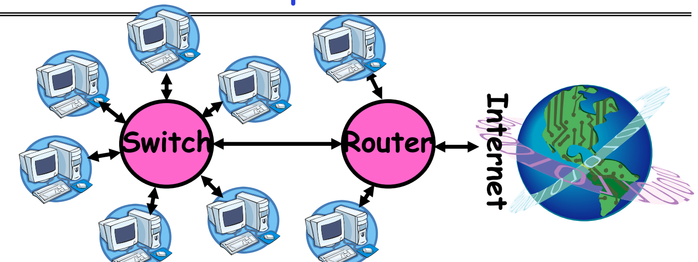


## Carrier Sense, Multiple Access/Collision Detection

- Ethernet (early 80's): first practical local area network
  - It is the most common LAN for UNIX, PC, and Mac
  - Use wire instead of radio, but still broadcast medium
- Key advance was in arbitration called CSMA/CD: Carrier sense, multiple access/collision detection
  - **Carrier Sense:** don't send unless idle
    - » Don't mess up communications already in process
  - **Collision Detect:** sender checks if packet trampled.
    - » If so, abort, wait, and retry.
  - **Backoff Scheme:** Choose wait time before trying again
- How long to wait after trying to send and failing?
  - What if everyone waits the same length of time? Then, they all collide again at some time!
  - Must find way to break up shared behavior with nothing more than shared communication channel
- Adaptive randomized waiting strategy:
  - **Adaptive and Random:** First time, pick random wait time with some initial mean. If collide again, pick random value from bigger mean wait time. Etc.
  - Randomness is important to decouple colliding senders
  - Scheme figures out how many people are trying to send!



## Point-to-point networks



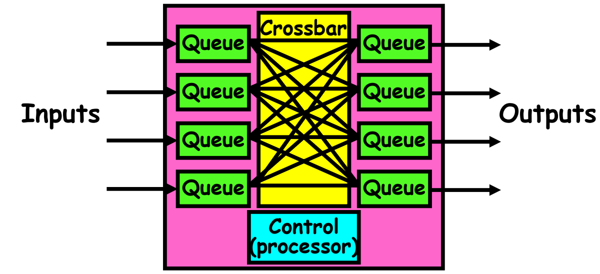

- Why have a shared bus at all? Why not simplify and only have point-to-point links + routers/switches?
  - Didn't used to be cost-effective
  - Now, easy to make high-speed switches and routers that can forward packets from a sender to a receiver.
- **Point-to-point network:** a network in which every physical wire is connected to only two computers
- **Switch:** a bridge that transforms a shared-bus configuration into a point-to-point network.
- **Router:** a device that acts as a junction between two networks to transfer data packets among them.

## Point-to-Point Networks Discussion

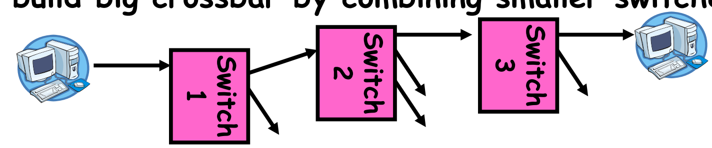
- Advantages:
  - Higher link performance
    - » Can drive point-to-point link faster than broadcast link since less capacitance/less echoes (from impedance mismatches)
  - Greater aggregate bandwidth than broadcast link
    - » Can have multiple senders at once
  - Can add capacity incrementally
    - » Add more links/switches to get more capacity
  - Better fault tolerance (as in the Internet)
  - Lower Latency
    - » No arbitration to send, although need buffer in the switch
- Disadvantages:
  - More expensive than having everyone share broadcast link
  - However, technology costs now much cheaper
- Examples
  - ATM (asynchronous transfer mode)
    - » The first commercial point-to-point LAN
    - » Inspiration taken from telephone network
  - Switched Ethernet
    - » Same packet format and signaling as broadcast Ethernet, but only two machines on each ethernet.



## Point-to-Point Network design

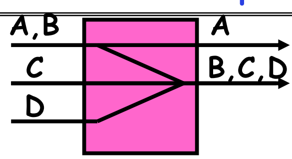



- Switches look like computers: inputs, memory, outputs
  - In fact probably contains a processor
- Function of switch is to forward packet to output that gets it closer to destination
- Can build big crossbar by combining smaller switches






## Flow control options

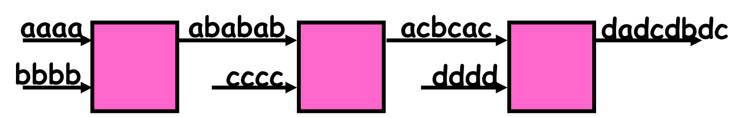



- What if everyone sends to the same output?
  - Congestion—packets don't flow at full rate
- In general, what if buffers fill up?
  - Need flow control policy
- Option 1: no flow control. Packets get dropped if they arrive and there's no space
  - If someone sends a lot, they are given buffers and packets from other senders are dropped
  - Internet actually works this way
- Option 2: Flow control between switches
  - When buffer fills, stop inflow of packets
  - Problem: what if path from source to destination is completely unused, but goes through some switch that has buffers filled up with unrelated traffic?



## Flow Control (con't)

- Option 3: Per-flow flow control.
  - Allocate a separate set of buffers to each end-to-end stream and use separate "don't send me more" control on each end-to-end stream




- Problem: fairness
  - Throughput of each stream is entirely dependent on topology, and relationship to bottleneck
- Automobile Analogy
  - At traffic jam, one strategy is merge closest to the bottleneck
    - » Why people get off at one exit, drive 50 feet, merge back into flow
    - » Ends up slowing everybody else a huge emount
  - Also why have control lights at on-ramps
    - » Try to keep from injecting more cars than capacity of road (and thus avoid congestion)

## Conclusion

- Buffer Cache: Memory used to cache kernel resources, including disk blocks and name translations
  - Read Ahead Prefetching: fetch sequential blocks early
  - Delayed Writes: Writes to files not immediately sent out to disk
- Important system properties
  - Availability: how often is the resource available?
  - Durability: how well is data preserved against faults?
  - Reliability: how often is resource performing correctly?
- RAID: Redundant Arrays of Inexpensive Disks
  - RAID1: mirroring, RAID5: Parity block
- VFS: Virtual File System layer
  - NFS: An example use of the VFS layer
- Network: physical connection that allows two computers to communicate
  - Packet: unit of transfer, sequence of bits carried over the network