Announcements

- Reading: Chapter 16
- Project #5 Due on Friday at 6:00 PM

Distributed Systems

- Provide:
  - access to remote resources
  - security
  - location independence
  - load balancing
- Basic Services:
  - remote login (telnet and rlogin protocols)
    - extends basic access provided by normal login
  - file transfer (ftp, rcp)
    - can support anonymous transfers
  - information services (http)
    - two way protocols (request/response)

Distributed Systems

- A unified view of local and remote access
- Typical Services
  - data migration
    - provide only the data required, not the whole file
    - manage multiple copies as versions of the same object
  - process migration
    - a process can move from one machine to another
    - reasons for migration:
      - load balancing
      - data affinity
      - hardware/software preference (better configuration)

Distributed OS Design Issues

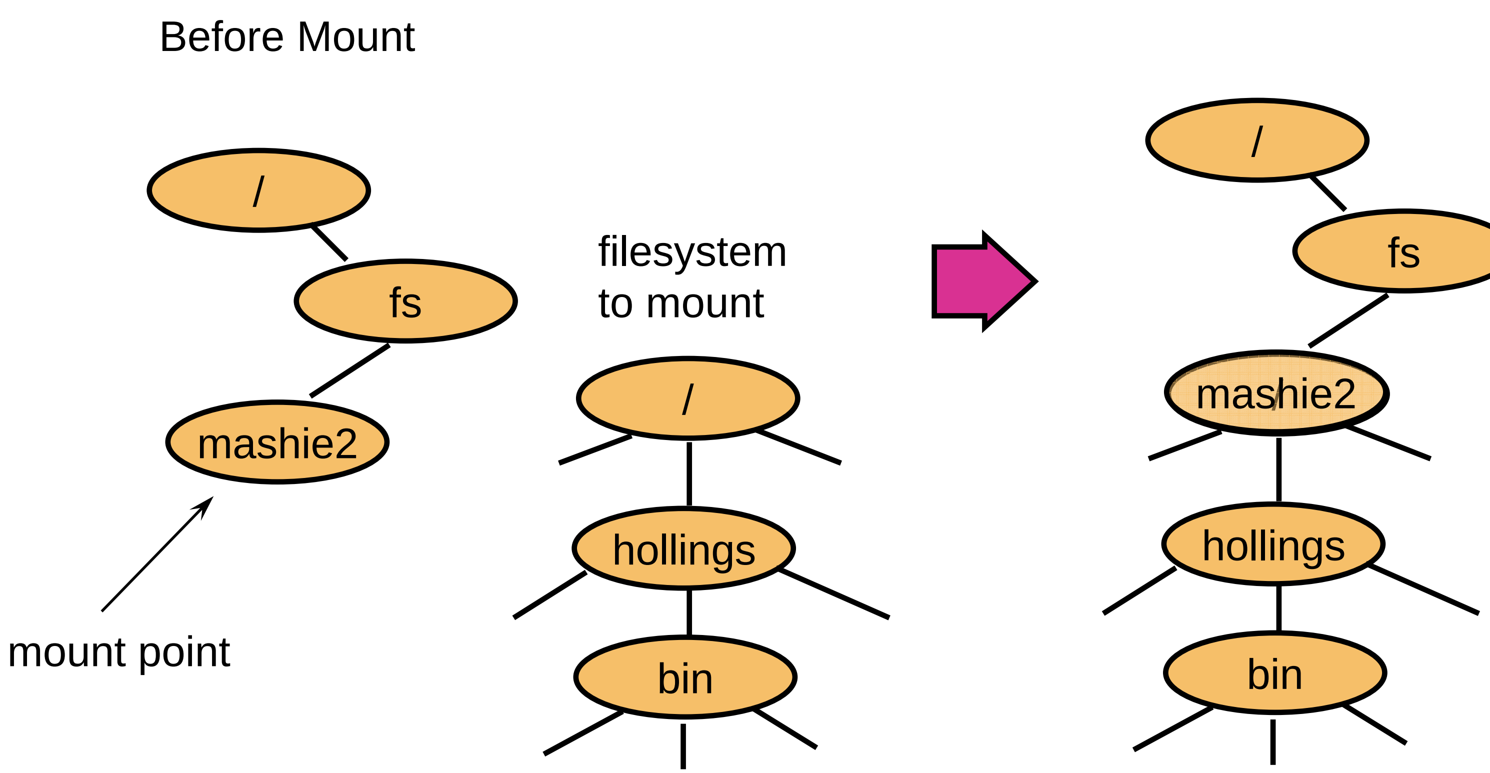
- Should provide same model as a central system
  - easy to understand for users
- Needs to be scaleable
  - will it work with 100, 1,000, or 10,000 nodes?
- Failure Modes
  - avoid a single central failure point
  - can loss performance or functionality with failure
    - but loss should be proportional to size of failure
- Security
  - should provide same guarantees on data integrity as a local system

File Server State

- Does the fileserver maintain information between requests?
- Stateless
  - example: NFS
  - each request contains a request to read/write a specific part of a file
  - requests must be *itempotent*
    - the same request can be applied several times
  - makes recovery of failed clients/servers easier
- Stateful
  - example: AFS
  - servers maintain connections for clients
  - improves performance
  - required for server based cache management

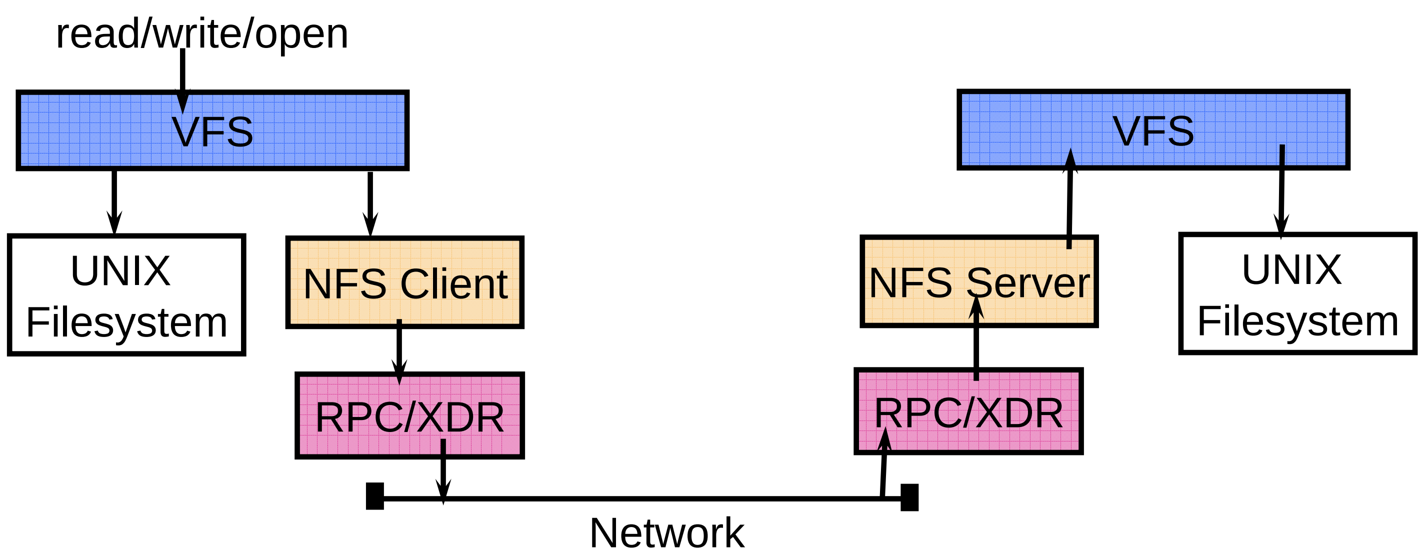
Mounting a filesystem

- Mount attaches a filesystem to a directory
  - can be used for local or remote (NFS) filesystems



NFS

- Provides a way to mount remote filesystems
  - can be done explicitly
  - can be done automatically (called an automounter)
  - clients are provided “file handle” by the server for future use
- Uses VFS: extended UNIX filesystem
  - inodes are replaced by vnodes
    - network wide unique inodes
    - can refer to local or remote files



NFS (cont.)

- Requests
  - are sent via RPC to the server
  - include read/write
  - query: lookup this directory info
    - must be done one step (directory) at a time
  - change meta data: file permissions, etc.
- Popular due to free implementations
- Provides no coherency

AFS

- Designed to scale to 5,000 or more workstations
- Location independent naming
  - within a single cell
- volumes
  - basic unit of management
  - can vary in size
  - can be migrated among servers
- names are mapped to “fids”
  - 96 bit unique id’s for a file
  - three parts: volume, vnode, and uniqidentifier
  - location information is stored in a volume to location DB
    - replicated on every server

AFS (cont.)

- File Access
  - open: file is transferred from server to client
    - very large files may only be partially transferred
  - read/write: performed on the client
  - close: file (if dirty) is written back to server
    - can fail if the disk is full
- Consistency
  - clients have callbacks
  - sever informs client when another client writes data
  - only applies to open operation
  - only requires communication when:
    - more than one client wants to write
    - one client wants to write and others to read

Announcements

- Reading: Chapter 16, 17
- Project #5 Due on Friday at 6:00 PM

Routing

- How does a packet find its destination?
  - problem is called routing
- Several options:
  - source routing
    - end points know how to get everywhere
    - each packet is given a list of hops before it is sent
  - hop-by-hop
    - each host knows for each destination how to get one more hop in the right direction
- Can route packets:
  - per session
    - each packet in a connection takes same path
  - per packet
    - packets may take different routes
    - possible to have out of order delivery

Routing IP Datagrams

- Direct Delivery:
  - a machine on a physical network can send a physical frame directly to another
  - transmission of an IP datagram between two machines on a single physical network does not involve routers.
    - Sender encapsulates datagram into a physical frame, maps destination IP address to a physical address and sends frame directly to destination
  - Sender knows that a machine is on a directly connected network
    - compare network portion of destination ID with own ID - if these match, the datagram can be sent directly
  - Direct delivery can be viewed as the final step in any datagram transmission

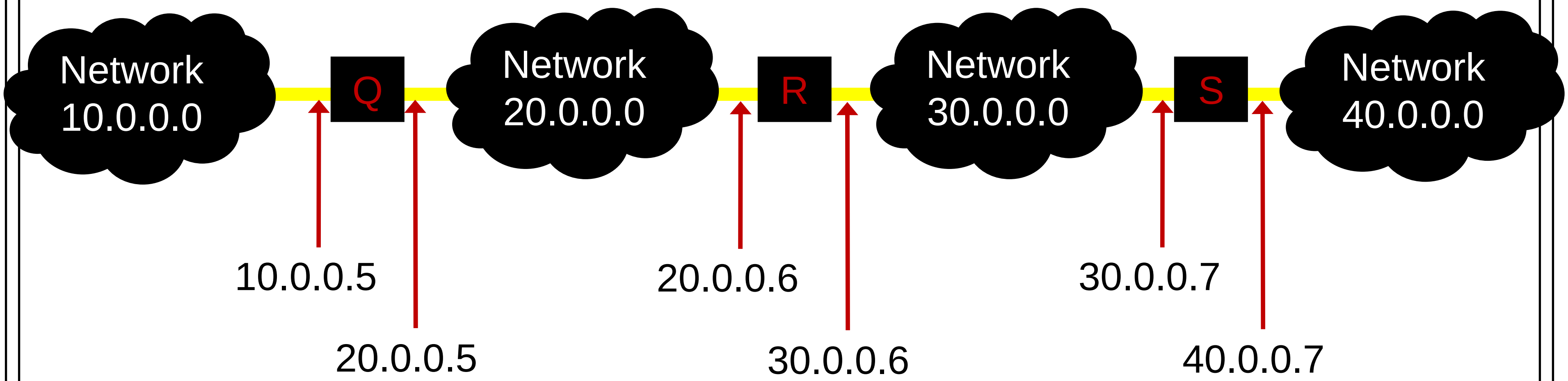
Routing Datagrams (cont.)

- Indirect Delivery
  - sender must identify a router to which a datagram can be sent
  - sending processor can reach a router on the sending processor's physical network (otherwise the network is isolated!)
  - when frame reaches router, router extracts encapsulated datagram and IP software selects the next router
    - datagram is placed in a frame and sent off to the next router

Table Driven Routing

- Routing tables on each machine store information about possible destinations and how to reach them
- Routing tables only need to contain network prefixes, not full IP addresses
  - No need to include information about specific hosts
- Each entry in a routing table points to a router that can be reached across a single network
- Hosts and routers decide
  - can packet be directly sent?
  - which router should be responsible for a packet (if there is more than one on physical net)

Routing



20.0.0.0	<DIRECT>
30.0.0.0	<DIRECT>
10.0.0.0	20.0.0.5
40.0.0.0	30.0.0.7

Example from Comer book: Internetworking with TCP/IP: volume 1 [Third Edition]

16

Algorithm: RouteDatagram (Datagram, RoutingTable)

Extract destination IP address, D, from datagram and compute network prefix N

If N matches any directly connected network address
  [Direct delivery]

Else if the table contains a host-specific route for D
  [send datagram to next-hop specified in table]

Else if the table contains a route for network N
  [send datagram to next-hop specified in table]

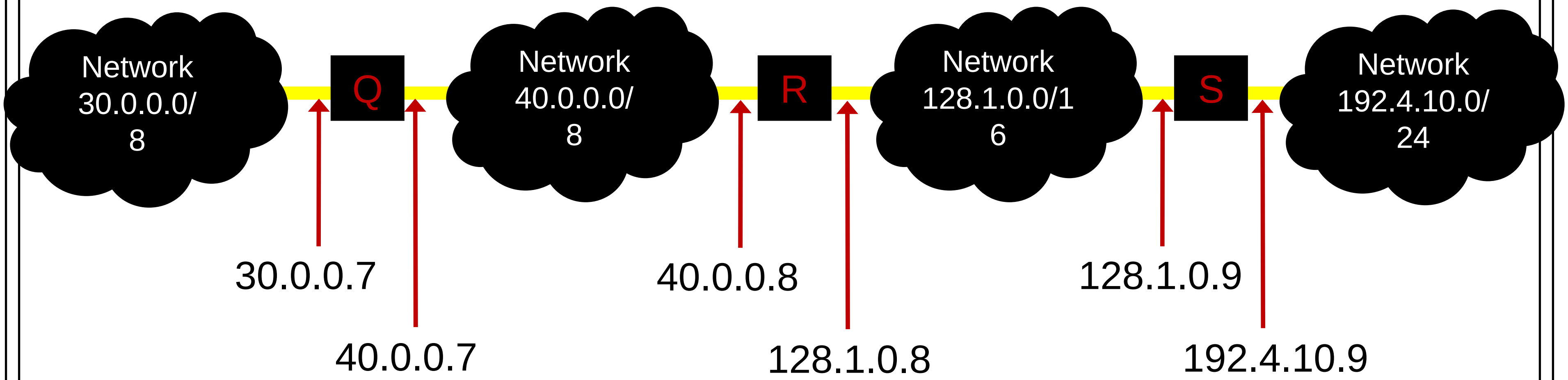
Else if the table contains a default route
  [send the datagram to the default route]

Else *declare a routing error*

Algorithm from Comer book: Internetworking with TCP/IP: volume 1 [Third Edition]

17

Routing (w/ subnets)



30.0.0.0	255.0.0.0	40.0.0.7
40.0.0.0	255.0.0.0	<DIRECT>
128.1.0.0	255.255.0.0	<DIRECT>
192.4.10.0	255.255.255.0	128.1.0.9

Mask field is used to extract the network part of an address during lookup.

If((Mask[i] & D) == Destination[i]) forward to nextHop[i]

Consider a datagram destined for address 192.4.10.3 and the datagram arrives at router R

Extract destination IP address, D from datagram and compute network prefix N

255.0.0.0&192.4.10.3 is not equal to 30.0.0.0

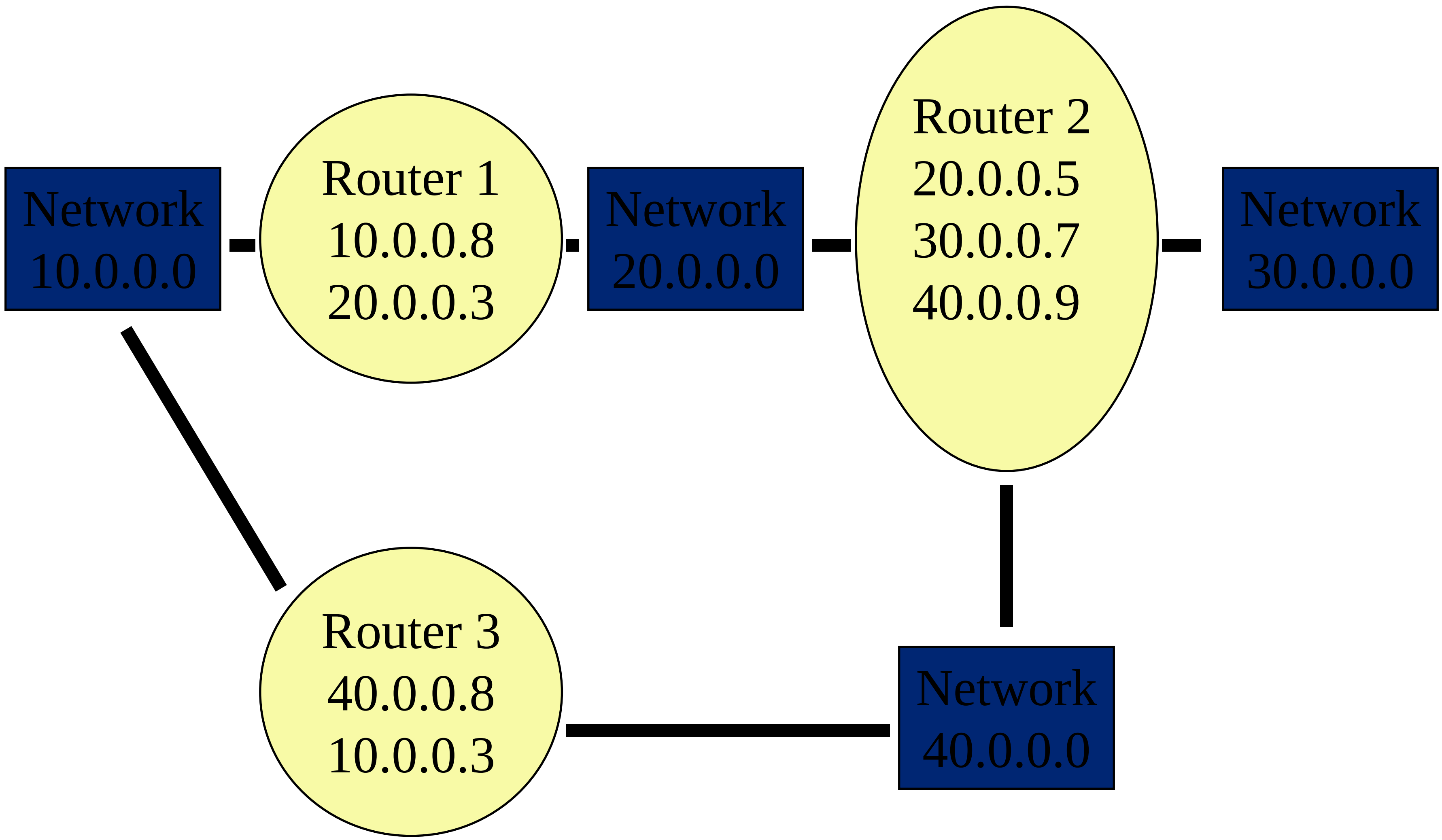
<same for entry 2 and 3>

255.255.255.0&192.4.10.3=192.4.10.0
→ send to 128.1.0.9

Example from Comer book: Internetworking with TCP/IP: volume 1 [Third Edition]

18

Routing


Network
10.0.0.0
Router 1
10.0.0.8
20.0.0.3
Network
20.0.0.0
Router 2
20.0.0.5
30.0.0.7
40.0.0.9
Network
30.0.0.0
Router 3
40.0.0.8
10.0.0.3
Network
40.0.0.0

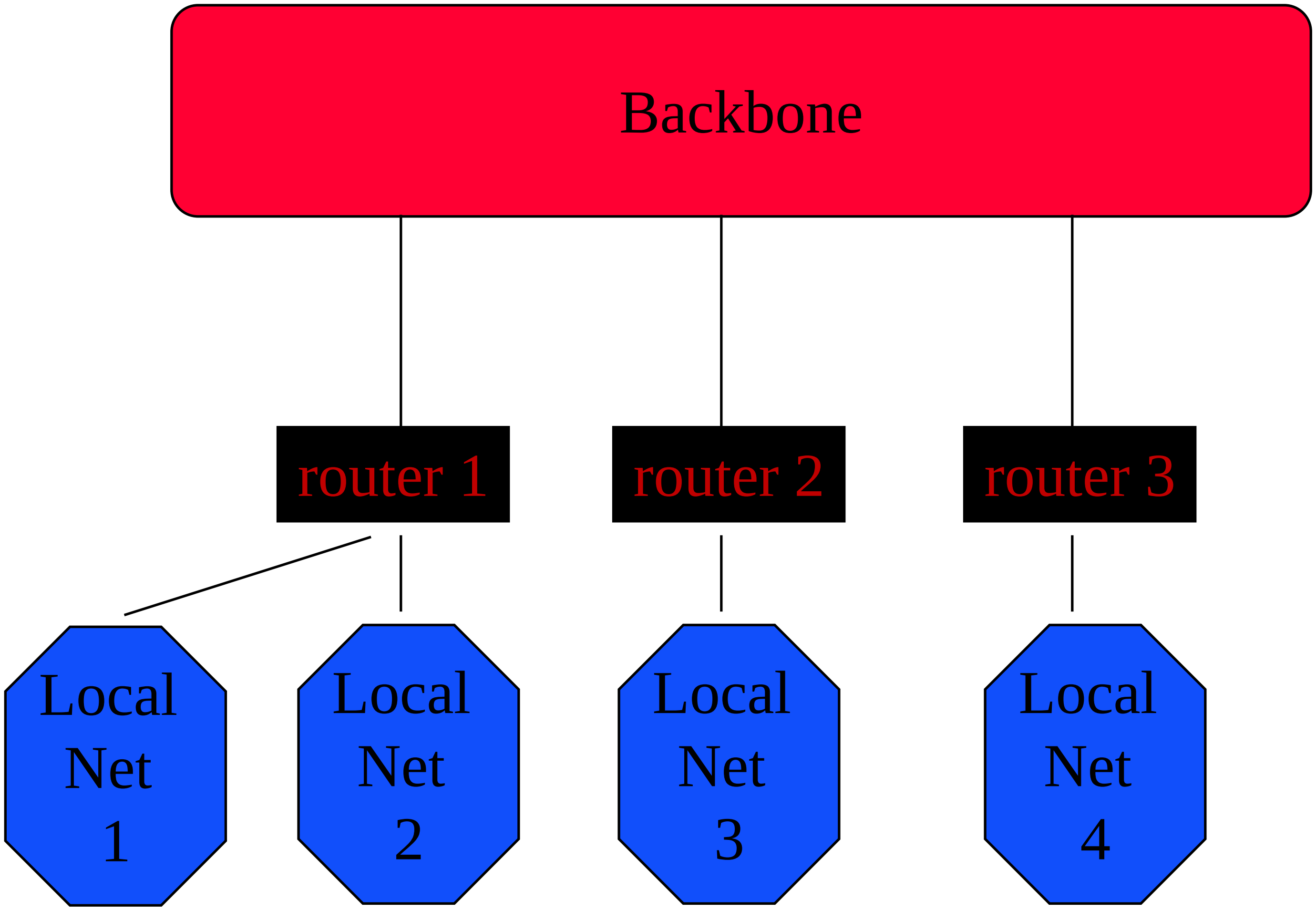
Routing with partial information

- Routing with partial information
  - Hosts do not need complete knowledge of all possible destination addresses
  - Host sends non-local information to (a) router
- Routers can also route with partial information
  - consider a topology consisting of two completely connected subgraphs A and B
  - subgraphs A and B share a single link
  - If a router in A sees an address it does not recognize, it sends the packet to B and vice-versa

Early Internet Architecture

- Small central set of routers that kept complete information about all destinations
- Larger set of outlying routers with only local information
- Default route for outlying routers is to a central router
- Local administrators can make changes
  - Local changes need to be propagated locally as well as to the central routers

Internet Core Router System



Internet Core Routing System

- Core routers exchange routing information so each will have complete information about optimal routes to all destinations
- This did not scale:
  - maintaining consistency among core routers became increasingly difficult
  - further difficulties arise when there are several backbones (e.g. ARPAnet and NSFnet)
  - if the core architecture is partitioned so that all routers use default routes, may induce routing loops
    - if routing information is not consistent, it is possible for a packet to be repeatedly routed in a circle until the packet times out

Distributed Systems

- Provide:
  - access to remote resources
  - security
  - location independence
  - load balancing
- Basic Services:
  - remote login (telnet and rlogin protocols)
    - extends basic access provided by normal login
  - file transfer (ftp, rcp)
    - can support anonymous transfers
  - information services (http)
    - two way protocols (request/response)

Distributed Systems

- A unified view of local and remote access
- Typical Services
  - data migration
    - provide only the data required, not the whole file
    - manage multiple copies as versions of the same object
  - process migration
    - a process can move from one machine to another
    - reasons for migration:
      - load balancing
      - data affinity
      - hardware/software preference (better configuration)

Distributed OS Design Issues

- Should provide same model as a central system
  - easy to understand for users
- Needs to be scalable
  - will it work with 100, 1,000, or 10,000 nodes?
- Failure Modes
  - avoid a single central failure point
  - can loss performance or functionality with failure
    - but loss should be proportional to size of failure
- Security
  - should provide same guarantees on data integrity as a local system

Distributed file systems

- Distributed systems can share physically dispersed files by using a distributed file system
  - Transparent DFS allows user mobility by bringing a user's environment (home directory) to wherever she logs in

- Naming: Location transparency vs. independence
  - Transparency: name does not hint on file's physical storage location (ex. NFS)
  - Independence: name of the file does not need to change when the file's physical storage location changes (ex. AFS)

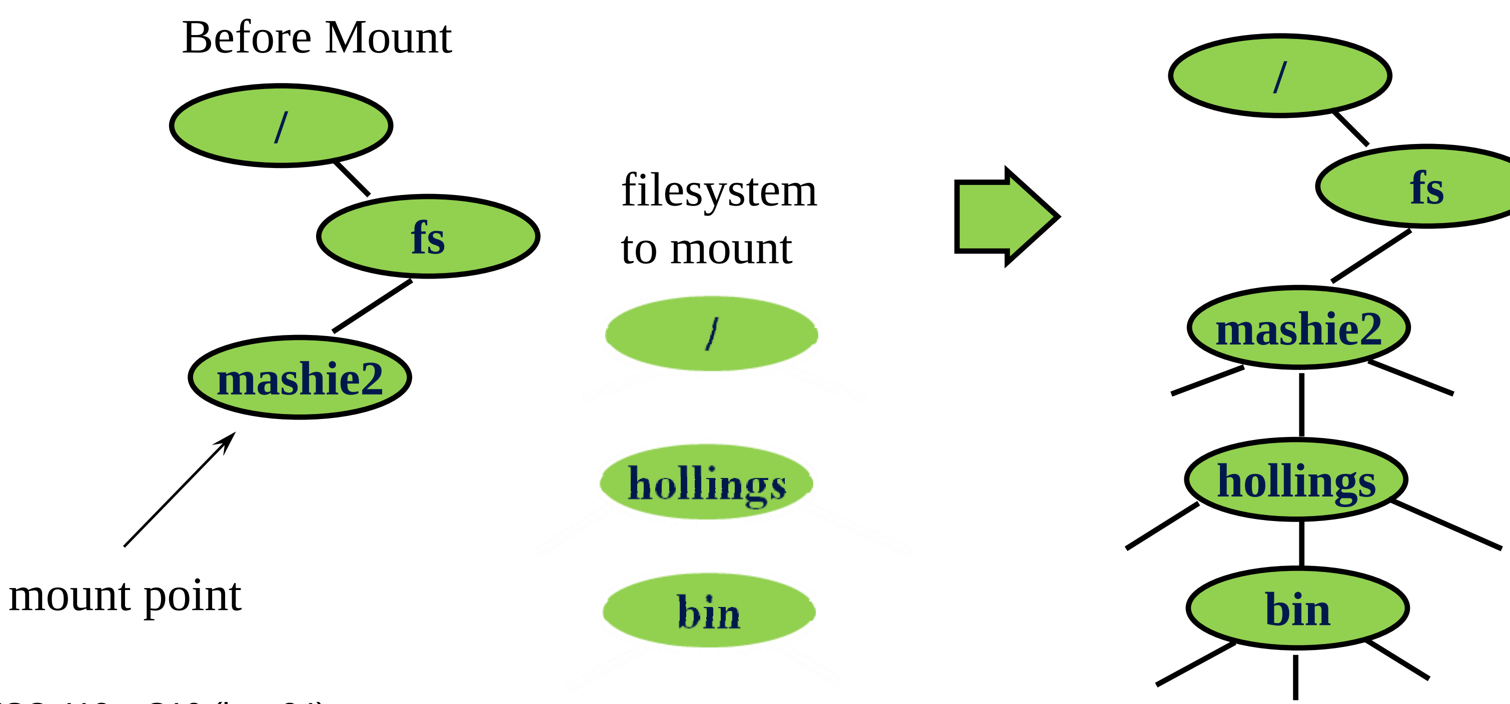
27

File Server State

- Does the fileserver maintain information between requests?
- Stateless
  - example: NFS (no open/close ops)
  - each request contains a request to read/write a specific part of a file
  - requests must be idempotent
    - the same request can be applied several times
  - makes recovery of failed clients/servers easier
- Stateful
  - example: AFS (explicit open/close ops)
  - servers maintain connections for clients
  - improves performance – via caching
  - required for server based cache management

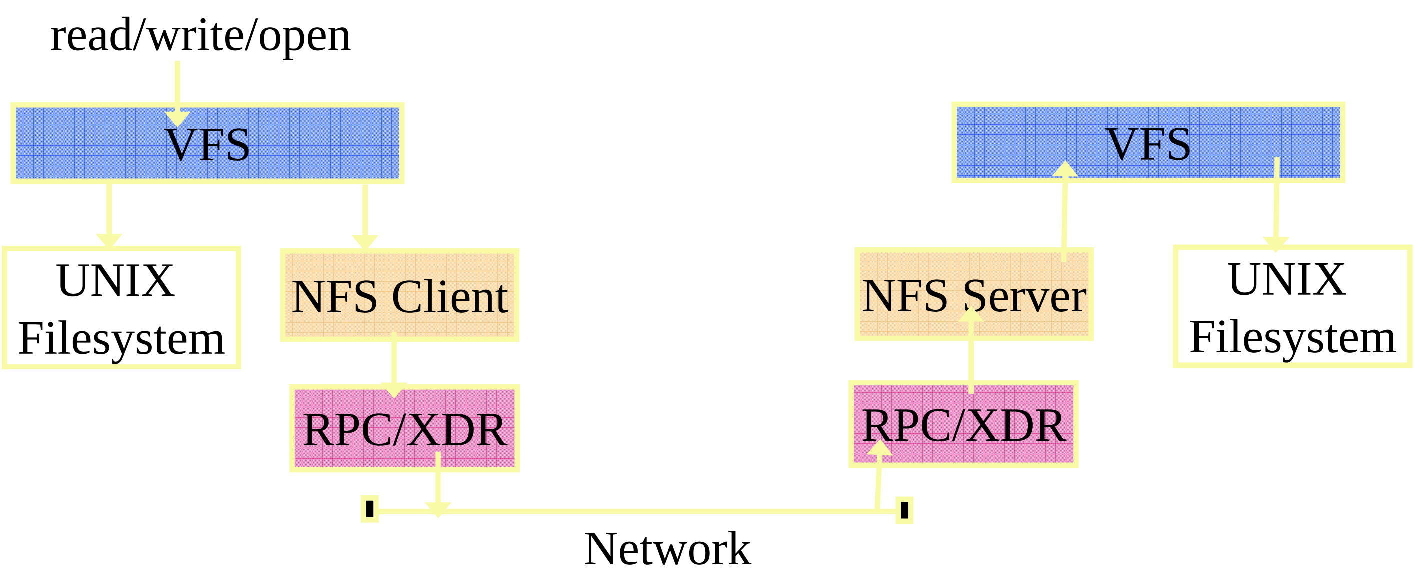
NFS: Mounting a filesystem

- Mount attaches a file-system to a directory
  - can be used for local or remote (NFS) file-systems



NFS

- Provides a way to mount remote file-systems
  - can be done explicitly
  - can be done automatically (called an automounter)
  - clients are provided "file handle" by the server for future use
- Uses VFS: extended UNIX file-system
  - inodes are replaced by vnodes
    - network wide unique inodes
    - can refer to local or remote files



NFS (cont.)

- Requests
  - are sent via RPC to the server
  - include read/write
  - query: lookup directory info
    - must be done one step (directory) at a time
  - change meta data: file permissions, etc.
- Popular due to free implementations
- Provides no coherency

AFS

- Designed to scale to 5,000 or more workstations
- Location independent naming
  - within a single cell
- volumes
  - basic unit of management
  - can vary in size
  - can be migrated among servers
- names are mapped to “fids”
  - 96 bit unique id’s for a file
  - three parts: volume, vnode, and uniqidentifier
  - location information is stored in a volume to location DB
    - replicated on every server

AFS (cont.)

- File Access
  - open: file is transferred from server to client
    - very large files may only be partially transferred
  - read/write: performed on the client
  - close: file (if dirty) is written back to server
    - can fail if the disk is full
- Consistency
  - clients have callbacks
  - sever informs client when another client writes data
  - only applies to open operation
  - only requires communication when:
    - more than one client wants to write
    - one client wants to write and others to read

Display and Window Management

- The screen is a resource in a workstation system
  - multiple processes desire to access the device and control it
  - OS needs to provide abstractions to permit the interaction
- Services
  - protection
  - windows
  - multiplex keyboard and mouse
  - configuration and placement
- Issues
  - how to get good performance and remain device independent
  - how much policy to dictate to users