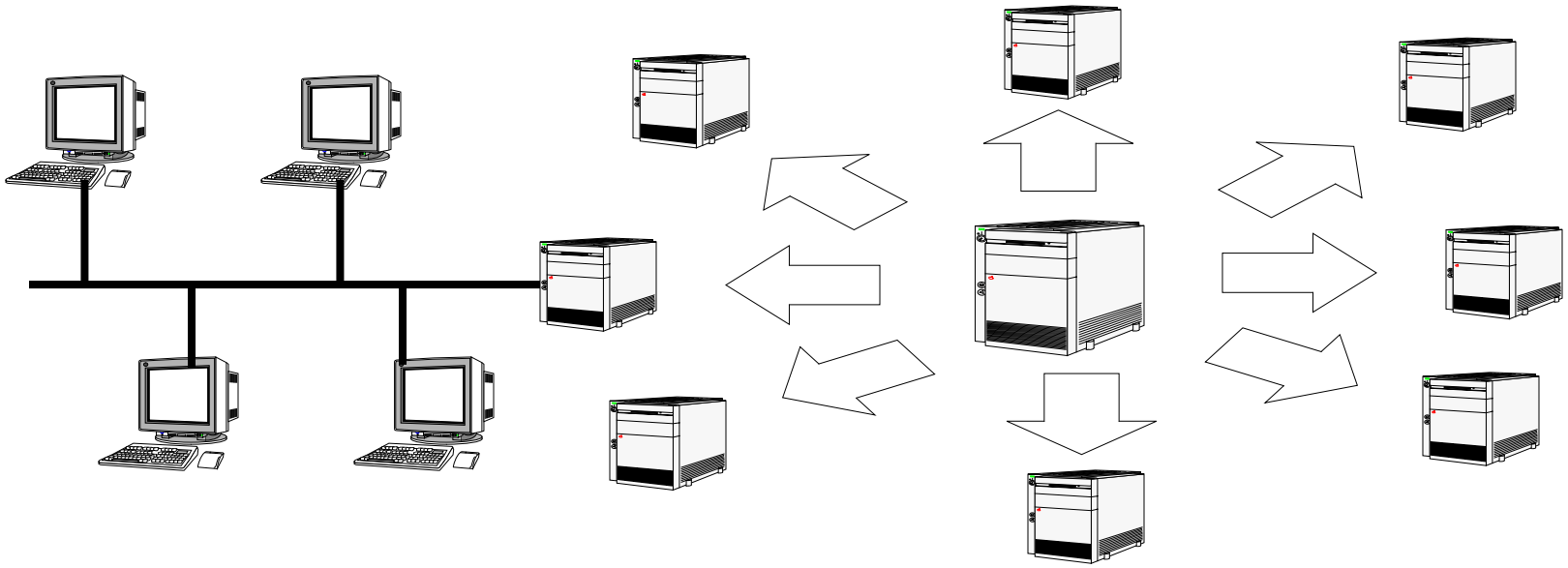
An NFS Replication Hierarchy

Brent Callaghan

Sun Microsystems, Inc

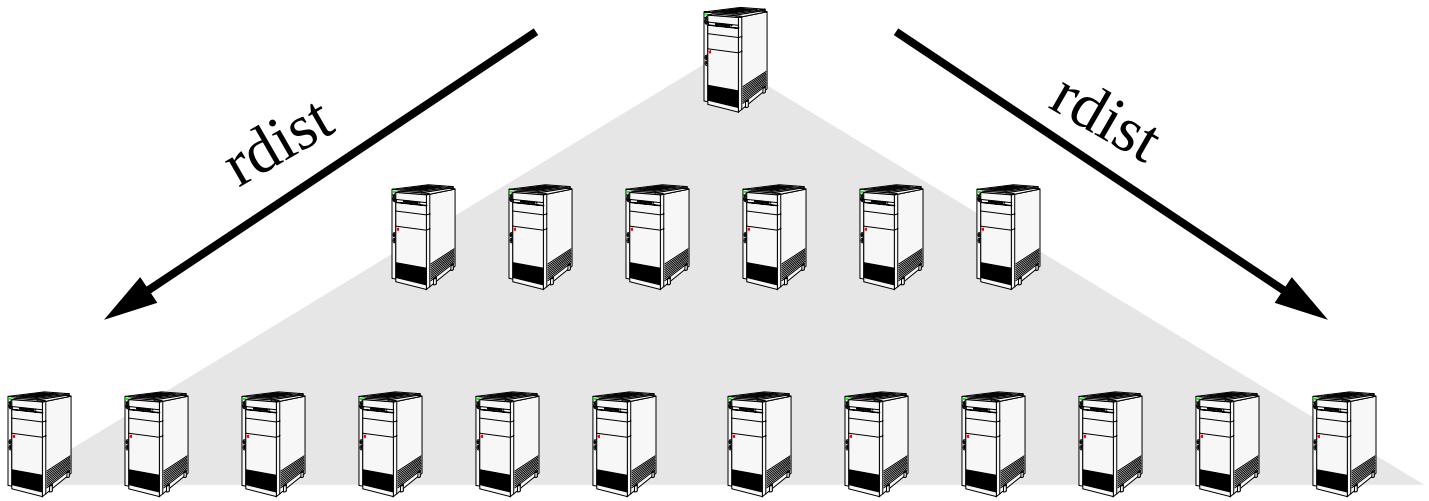
The Problem: /usr/local

- Software packages and other shared data
- Replication for high availability, scalability, network traffic
- Support for heterogeneous systems
- Consistency for failover.



Example: Sun's "/usr/dist"

- **In each replica:**
  - **11 Gigabytes of data – 300,000 files**
  - **150 packages**
- **400 Servers worldwide in 6 level hierarchy**
- **Current distribution daily via hierarchical** rdist

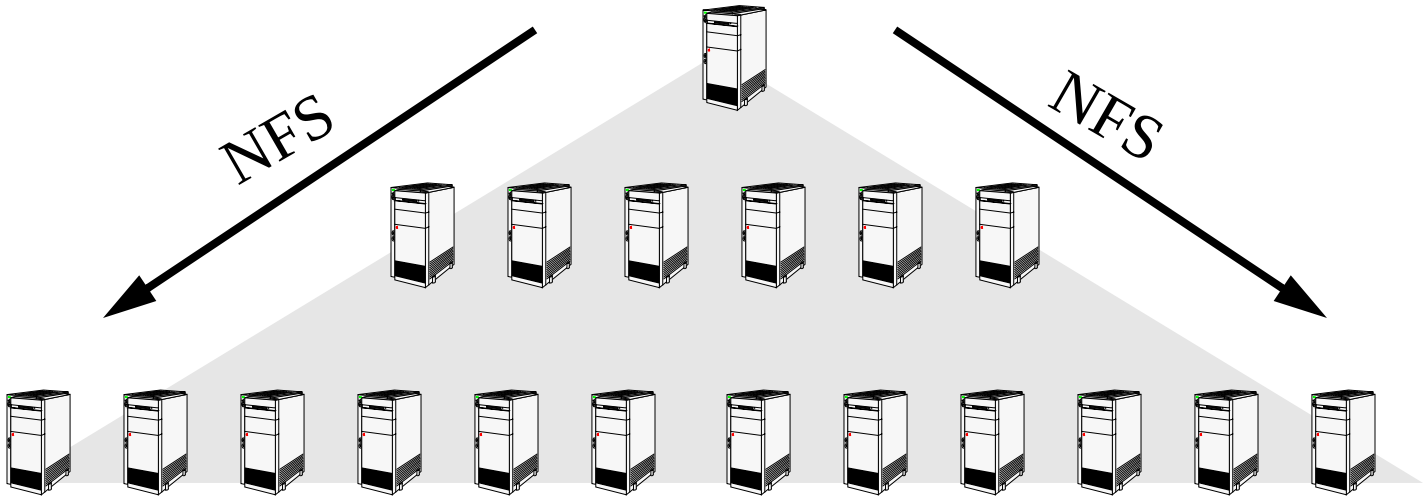


Problems with rdist

- Complex setup
- Clients must register with distribution server
- Point-to-point design (no hierarchical support)
- Unix only
- Recovery can be difficult
- No support for differential update
- Push rather than pull
- Other alternatives:
  - mirror (ftp via Perl script)
  - rsync (rsh)

NFS Hierarchy Overview

- Nodes are NFS clients/servers
- Data “pulled” down hierarchy by polling for changes
- Changes made only at master node
- End nodes are diskless clients or replicas themselves.

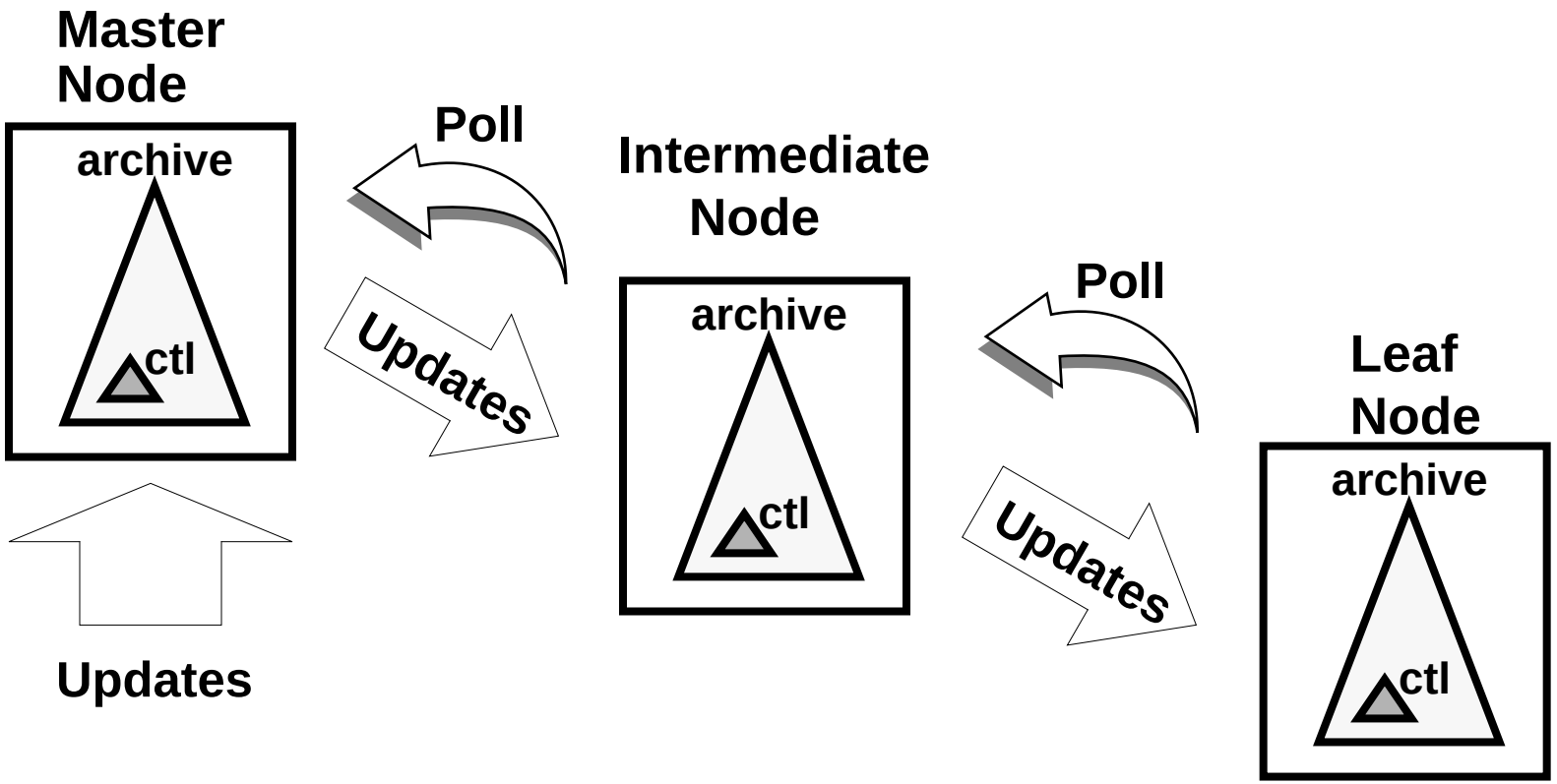


Advantages of NFS

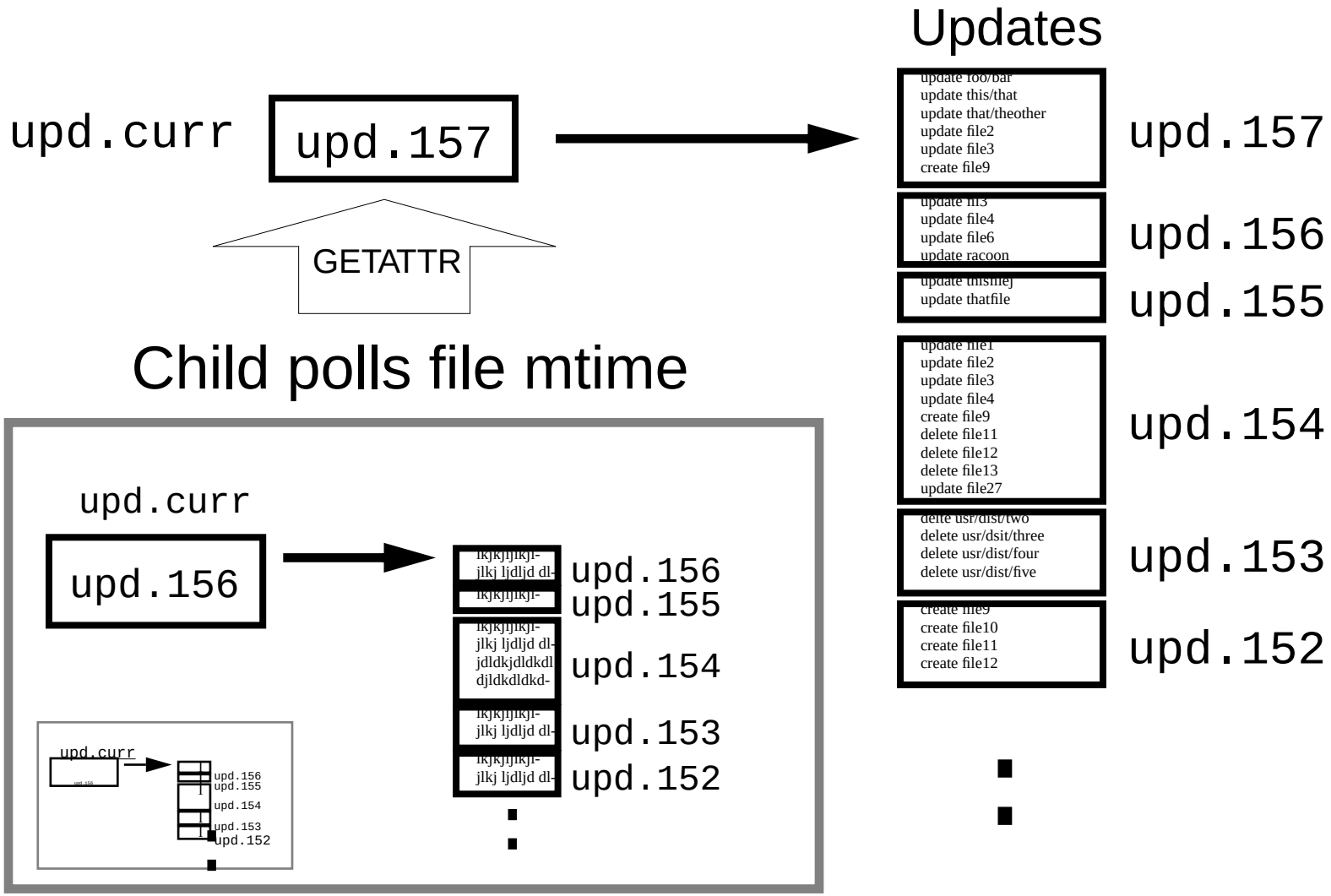
- **High capacity, kernel-resident server — good performance!**
- **Concurrent service to multiple clients**
- **Easier setup & configuration**
- **Resistant to network/node failure *"NFS server not responding"***
- **Protocol designed for remote access to file hierarchies**
  - **Access to directories, files, symbolic links, hard links**
- **NFS already installed or readily available from multiple vendors.**
- **Administrators already familiar with NFS**
- **/usr/local servers are already NFS servers.**

Replication Hierarchy Detail

- Replication daemon runs at intermediate and leaf nodes
- Initial setup via network copy or tape
- Thereafter maintenance by automatic network updates



Control Directory



Update Files (contd)

- Child polls parent's `ctl/upd.curr` file for changed mtime
- When updates complete, child sets `upd.curr` mtime to parent's.
- Configuration is *transitive* – when updates complete the child's `archive` (incl `ctl`) directories are identical to parent's
- Child may use several updates to "catch up" due to:
  - Network outage
  - Disconnection/reconnection (mobile computer)

Update File Syntax

- **Operations to bring replica up to date, one per line. All pathnames relative to archive root.**

- **`update`** *pathname*
  - **Create or update a regular file, directory or symbolic link**
- **`delete`** *pathname*
  - **Remove file, directory (& subdirs) or symbolic link**
- **`rename`** *pathname1 pathname2*
  - **Rename a file, directory or symlink**
- **`link`** *pathname1 pathname2*
  - **Create a new hard link**

- **copy** *pathname1 pathname2*
  - **Make a copy of a file or directory**
- **exec** *command args ...*
  - **Execute local command**

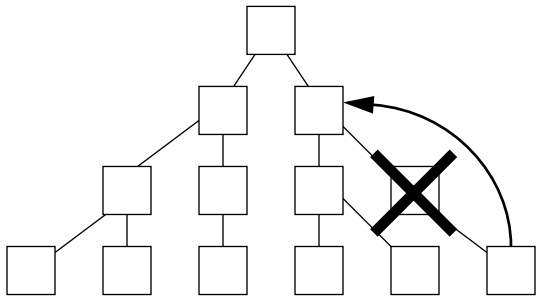
Generating Updates

- Comparing client with server hierarchy can be expensive:
  - Walk /usr/dist tree with “find”: 17 minutes (mtimes & sizes)
  - Read all files: 105 minutes (checksums)
  - Repeated for each client
  - Used by rdist, rsync
- Compute once at server
  - Take snapshot
  - Make changes
  - Compare hierarchy with snapshot
  - Generate list of created, updated, deleted & renamed files
  - Add manual updates

Partial Update

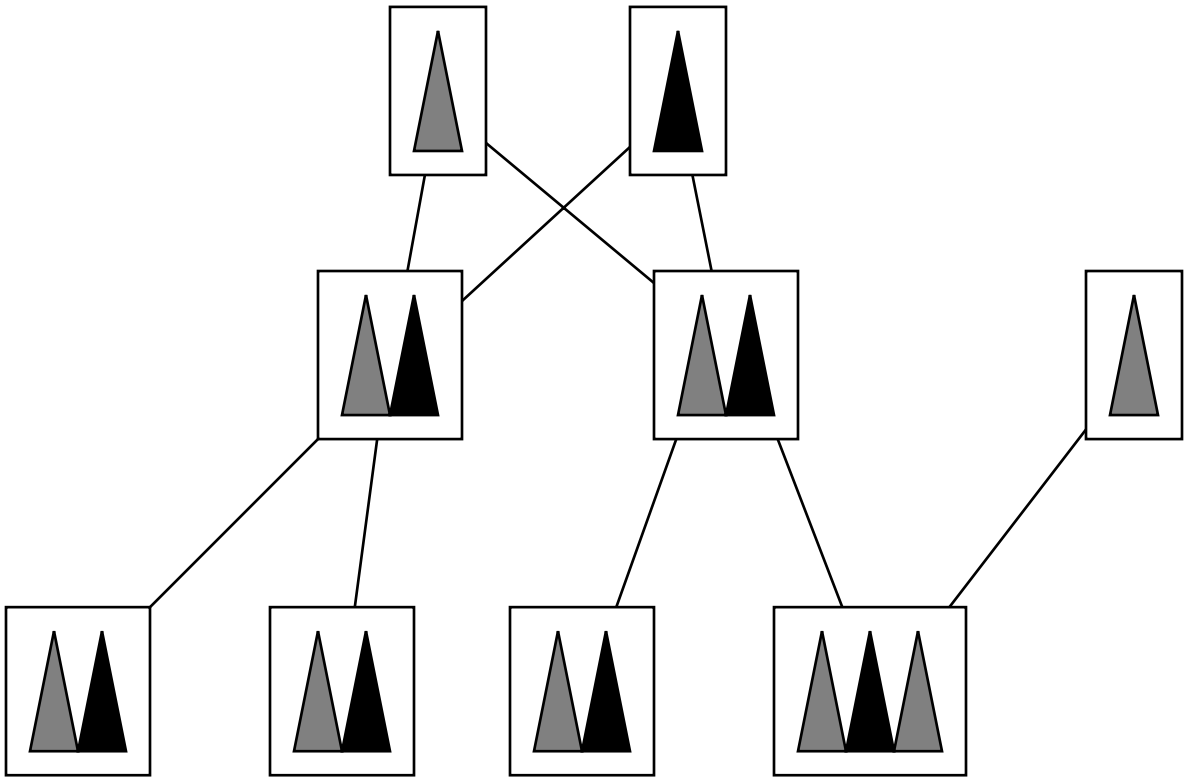
What if node loses contact with parent during update ?

- **Updates are non-idempotent - cannot be restarted.**
- **NFS is well known for dogged persistence.**
  Just keep trying until server responds and resume update
- **Assumption that update time << update frequency**
- **Can configure child to failover to grandparent.**



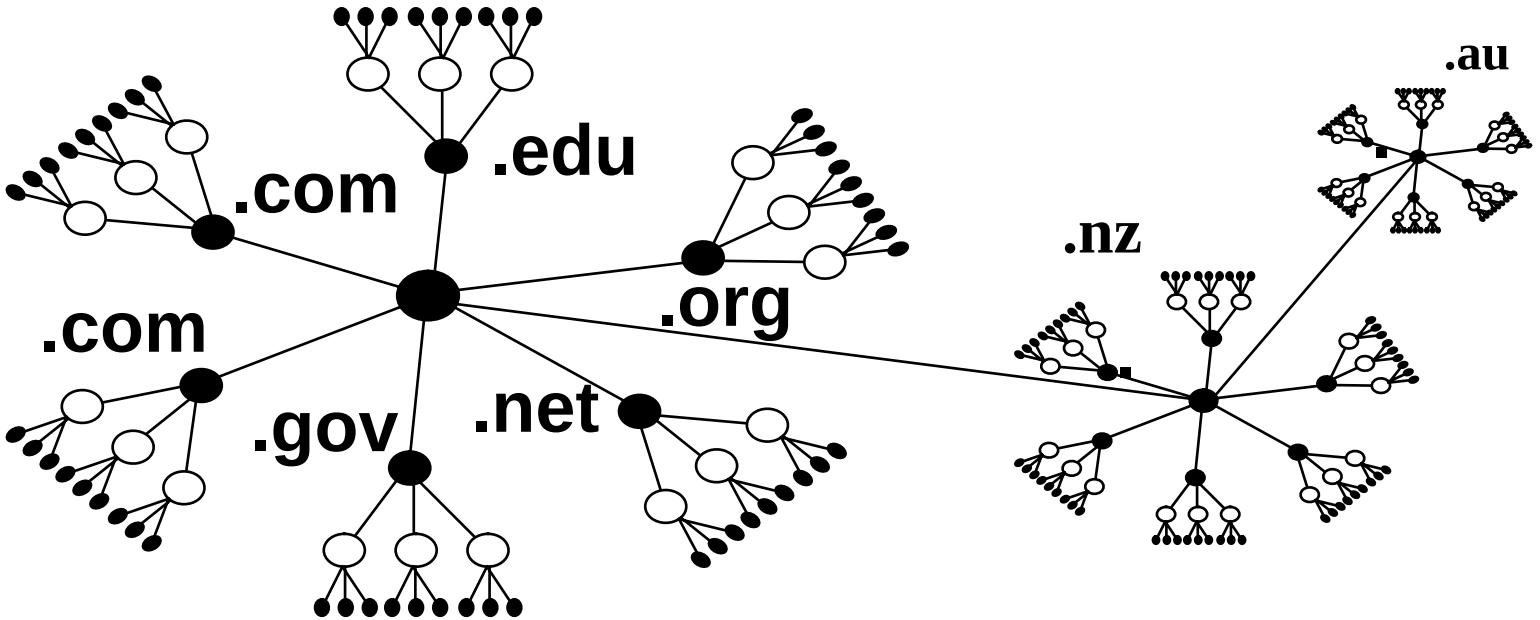
Multiple Hierarchies

- **Nodes can host multiple hierarchies**
  - **Each disjoint, configured independently**



An Internet Hierarchy

- Internet distribution using NFS over TCP
- All servers export read-only
- Servers have no knowledge of clients



Why NFS ?

- Could build a hierarchy with SMB, AFS or DCE/DFS or any distributed filesystem that you have lying around the house.
- Easy setup for sites that already have NFS servers.
- NFS servers handle high loads with good response time
  - Specbench SFS 93: max 27,862 ops/sec @ 18ms (rw)
- NFS clients recover automatically from server crash or lost TCP connection.
- Can use NFS v2, though V3 provides larger transfers, piggyback attributes, READDIRPLUS – better where latency is high.

NFS Security ?

- **General use of “trusted host” on Intranets**
  - `ro=engineering,rw=admin1`
- **Used on Internet to export “public” data readonly**
- **Could use secure tunnels on Internet**
- **RPC security: Diffie-Hellman & Kerberos v4 key exchange**
  - **Not widely available**
- **RPCSEC-GSS**
  - **IETF Working Group**
  - **Implements pluggable security based on GSS-API**
  - **Authentication, Integrity, Privacy**

Work in Progress

- **Still a prototype - not yet deployed within Sun**
- **Need to meet all requirements of rdist system:**
  - **Update logging**
  - **Error reporting**
  - **Customized installations**
  - **In-place updates**
- **Requirement for stable server “snapshot”**
  - **When is it safe for client to download updates ?**
  - **When is it safe for server to download updates ?**
- **Automatic generation of updates on master**
  - **Compare one snapshot with another and emit updates, renames, deletes**
  - **Merging automatic updates with manual updates**