DATA STORAGE TECHNOLOGIES & NETWORKS

(CS C446, CS F446 & IS C446)

LECTURE 30 – STORAGE

What is NFS?

- A remote file system protocol
  - Designed and implemented by Sun Microsystems
  - Provides UNIX-like file system interface and semantics
- It is implemented as a Client-Server application
  - Client part imports filesystems (from other computers)
  - Server part exports (local) filesystems
    - i.e. makes it visible on the network and enables access from other computers

NFS – Design Goals

- Stateless protocol
  - No state to maintain or recover
  - Can continue to operate even during the periods of client or server failure
    - Increased robustness – easy recovery from failures
- Support for UNIX file system semantics
  - Also supports other semantics such as MS-DOS
- Protection and Access Control same as UNIX file system semantics
- Transport-independent protocol
  - Originally implemented on top of UDP datagram protocol
  - Easily moved to TCP stream protocol later
  - Has been implemented over many non-IP-based protocols.

NFS - Design Limitations

- Design for clients and servers connected on a locally fast network.
  - Does not work well over slow links nor between clients and servers with intervening gateways.
  - Works poorly for mobile computing that has extended periods of disconnected operation
- Stateless protocol
  - Caching is an implementation detail – based on the assumption “most files will not be shared”
  - Semantics differs – in some cases – from traditional Unix semantics
    - E.g. locking (flock) implemented by a stateful daemon

NFS – Transport

- Client – Server Protocol (using a Request – Response model)
  - Server receives RPC – Remote Procedure Call – requests from clients
    - RPC operates like a local procedure call
  - Can be run on top of stream (e.g. TCP) or datagram protocol (e.g. UDP)
    - Each RPC message may need to be broken into multiple packets to be sent across the network.
      - May typically require up to 6 packets
      - May cause problems in UDP – in case of failure, entire message must be retransmitted

NFS – Transport

- Remote Procedure Call (RPC)
  - Client sees a procedure call interface (akin to a local procedure call)
  - But the call (along with the parameters) is *marshalled* into a message and sent over the network
    - *Marshalling* may involve replacing pointers by data to which they point to and converting binary data to the canonical network byte order
  - Server *unmarshalls* the message (i.e. separates it into pieces) and processes it as a local operation.
  - The result is similarly *marshalled* by the server and sent to the client which in turn *unmarshalls* it.

NFS – Interface

- NFS RPC requests
  - Idempotent Operations – an operation that can be repeated several times without the final results being changed or an error being caused
    - GETATTR, SETATTR, LOOKUP, READLINK, READ, WRITE, CREATE, SYMLINK, READDIR, STATFS
  - Non-Idempotent Operations
    - REMOVE, RENAME, LINK, MKDIR, RMDIR
- Idempotency is significant because of
  - slow links and lost RPC *acks*.
- Each file on the server is identified by a globally unique file handle

  created by the server when a pathname translation request (lookup) is sent by client to server

- Lookup
  - Server must find the requested file/directory with access permission
  - If granted, the server returns a file handler to the client
  - In NFS implementation, file handler is built from file system identifier, an inode number and a generation number
    - Server creates a unique file system identifier for each of its locally mounted file systems
    - Generation number is assigned to an inode each time the inode is allocated to represent a new file
    - Each generation number is used only once [usually uses a random number generator [or creation time of the file] to generate this]

- File system identifier and inode provide unique identifier for the inode to be accessed
- Generation number verifies that the inode still references the same file that it referenced
  - Generation number detects when the file has been deleted and a new file has been created later with the same inode number
  - Generation number ensures that the file handle is time stable
    - Time stable identifier in distributed file system is the one that refers uniquely to some entity both while that entity exists and for a long time after it is deleted
    - Time stable identifier allows a system to remember an identity across transient failures