SOLARIS NFS/TCP

Mike Eisler

mre@Eng.Sun.Com

- **Motivations**
- **Requirements**
- **Design**
- **Implementation War Stories**
- **Future Work**

MOTIVATIONS IN 1993

- Perceived higher demand for WAN usage
- Dynamic retransmission, timeout, and transfer re-sizing with NFS/UDP never worked well
- NFS protocol Version 3 clients and servers might want big RPC requests and replies

MOTIVATIONS IN 1996

- Internet explosion. Ftpd and httpd can’t scale like NFS servers can.
- Firewalls are kinder to TCP than UDP
- Public NFS concept

REQUIREMENTS

- **Interoperability with other implementations**
  - Interesting problem given that there is no actual specification for NFS protocol operation over TCP
- **No semantic changes from NFS/UDP**
- **Don’t compromise NFS system’s simple recovery**
- **No “unacceptable” performance drop from NFS/UDP on 10 mbit ethernet.**
- **Preserve support for NFS/UDP**
- **Support NFS Versions 2 and 3.**
- **Provide connection-oriented RPC support in kernel for other applications.**

DESIGN

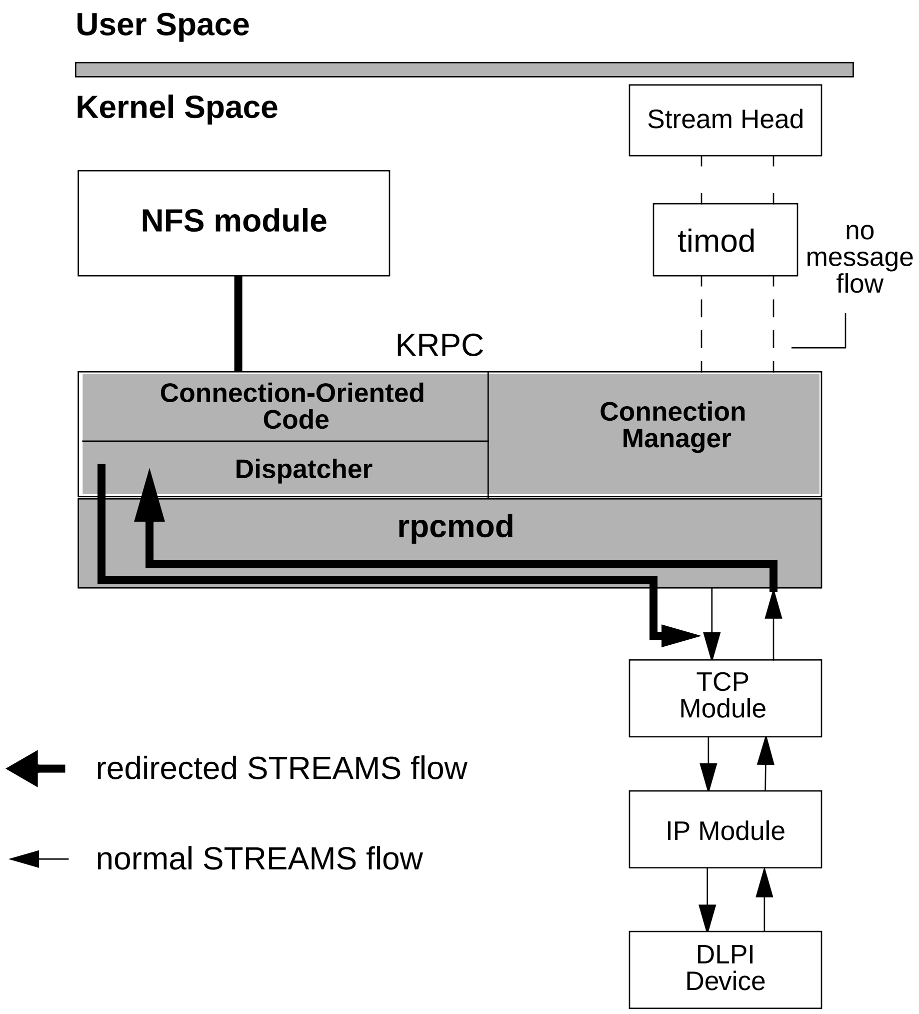
- **Overview**
  - Administration
  - Components
  - Record Marking
  - Idle Timers
- **Client Side**
  - Call semantics
  - Connection management
- **Server Side**
  - Connection management
  - Duplicate request cache

ADMINISTRATION

- **nfsd will by default listen over TCP and UDP**
    - there are options to limit operation over a specific protocol
- **mount command prefers to use TCP over UDP**
- **mount command (and automounter maps) take a “proto=protocol” suboption.**
- **nfsstat -m prints protocol selected:**

```
/net/dfs-10/export1 from dfs-10:/export1
 Flags:
vers=3,proto=tcp,sec=des,hard,intr,grpid,l
ink,symlink,acl,rsize=32768,wsize=32768
```

CLIENT SIDE COMPONENTS



User Space

Kernel Space

Stream Head

NFS module

timod

no message flow

KRPC

Connection-Oriented Code

Dispatcher

Connection Manager

rpcmod

TCP Module

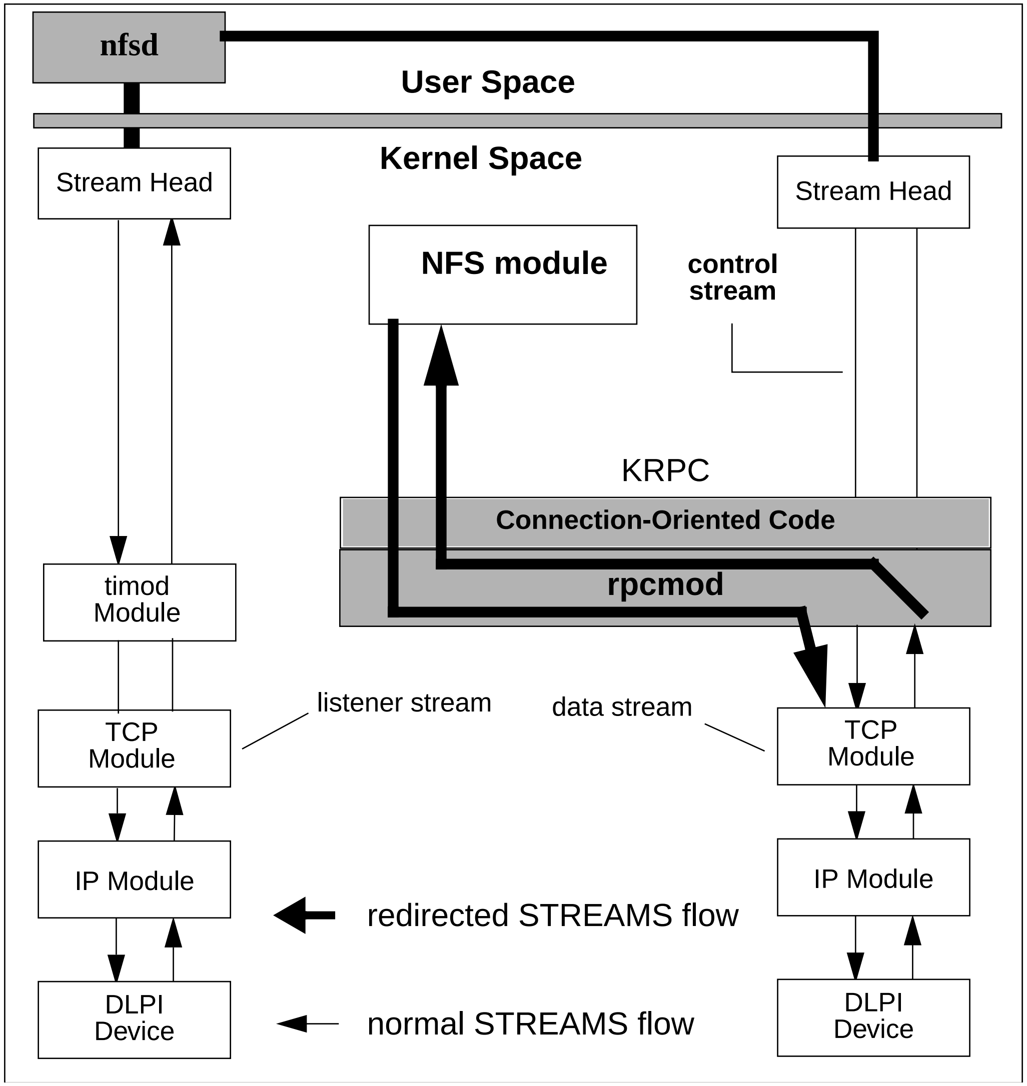
IP Module

DLPI Device

⬅ redirected STREAMS flow

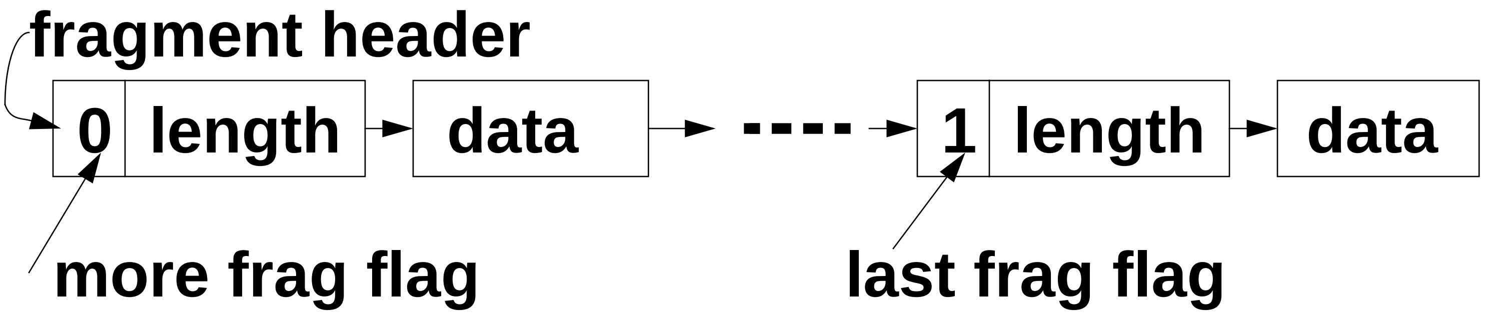
← normal STREAMS flow

SERVER SIDE COMPONENTS



RECORD MARKING

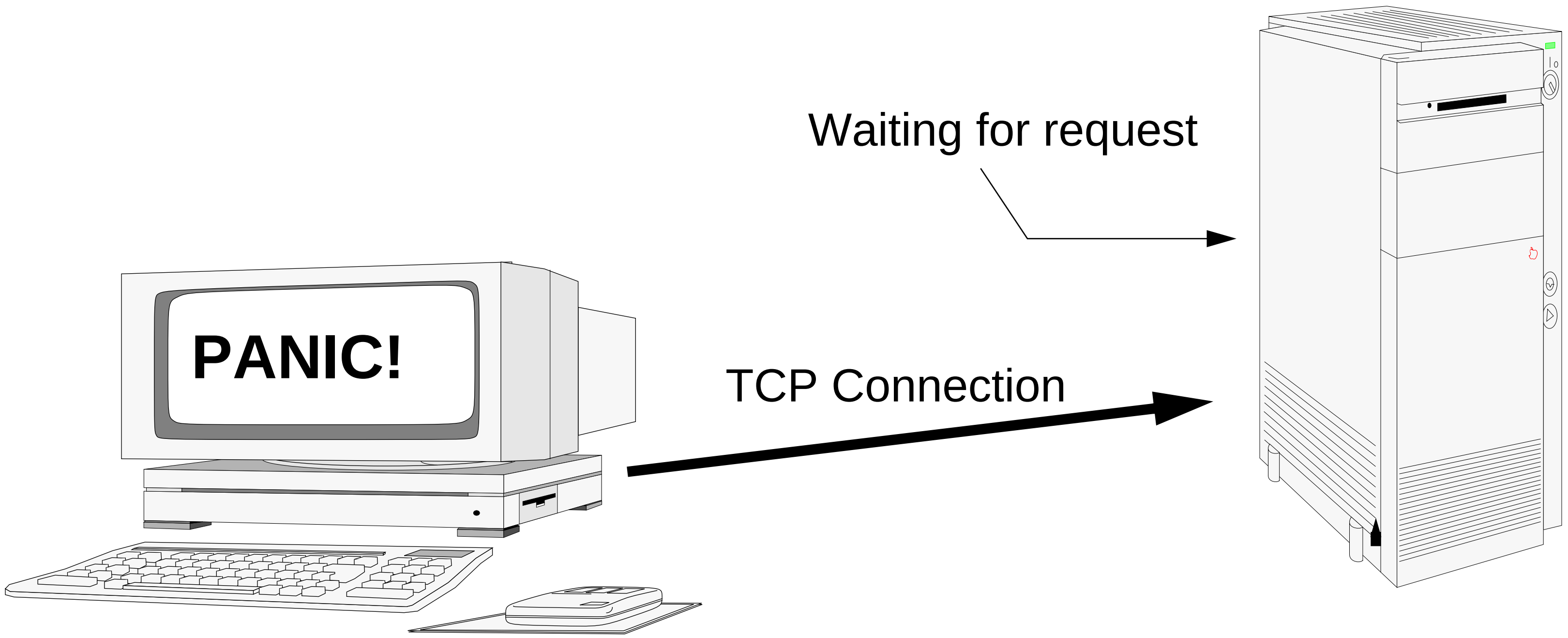
- ONC RPC method for putting records on byte stream transports:



- Transmitted records use single fragment
- Received fragments are gathered in rpcmod, then sent as one assembled record to kRPC logic.
- Bad lengths handled by disconnecting.

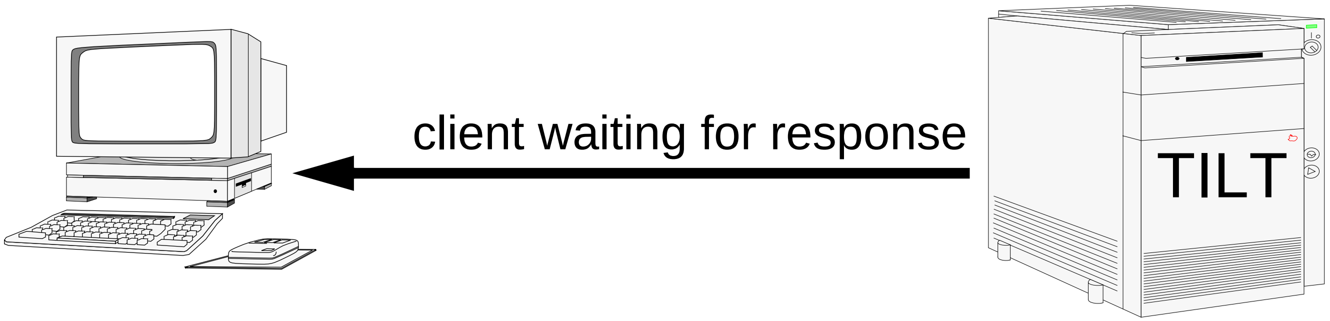
IDLE TIMERS

- **Idle connections on client and server are killed**
- **Client has 5 minute timer**
- **Server has 6 minute timer**
  - eliminates potential connection leak:



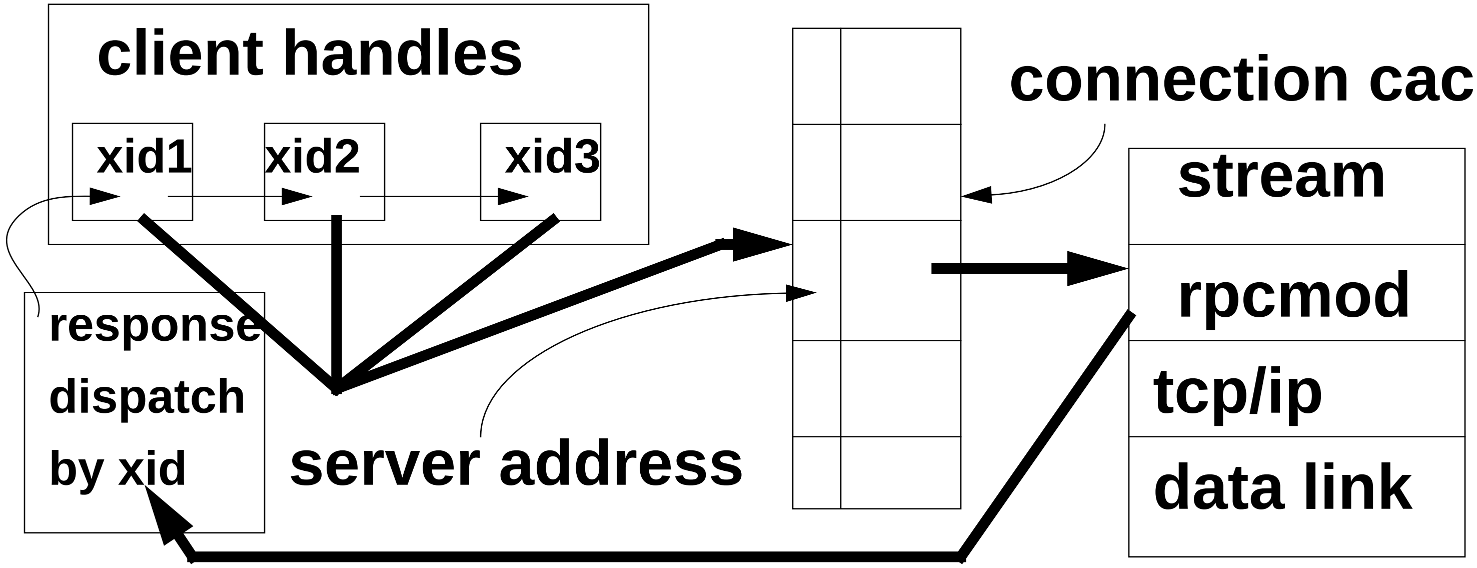
CALL SEMANTICS

- **CLNT_CALL() sends one RPC level request**
  - hard mounts try again, soft mounts stop.
- **Call can fail due to:**
  - failure to create connection:
    - **connection refusal (delay a bit before returning)**
    - **connection timeout**
  - failure to get reply:
    - **broken connection**
    - **call timeout. Even with reliable connection this is needed:**



CLIENT CONNECTION MANAGEMENT

- **Model is a fixed number of connections between client and server pair (default 1).**
- **Connections created on demand & cached after use**
- **Multiple client handles multiplex traffic over single connection:**



SERVER CONNECTION MANAGEMENT

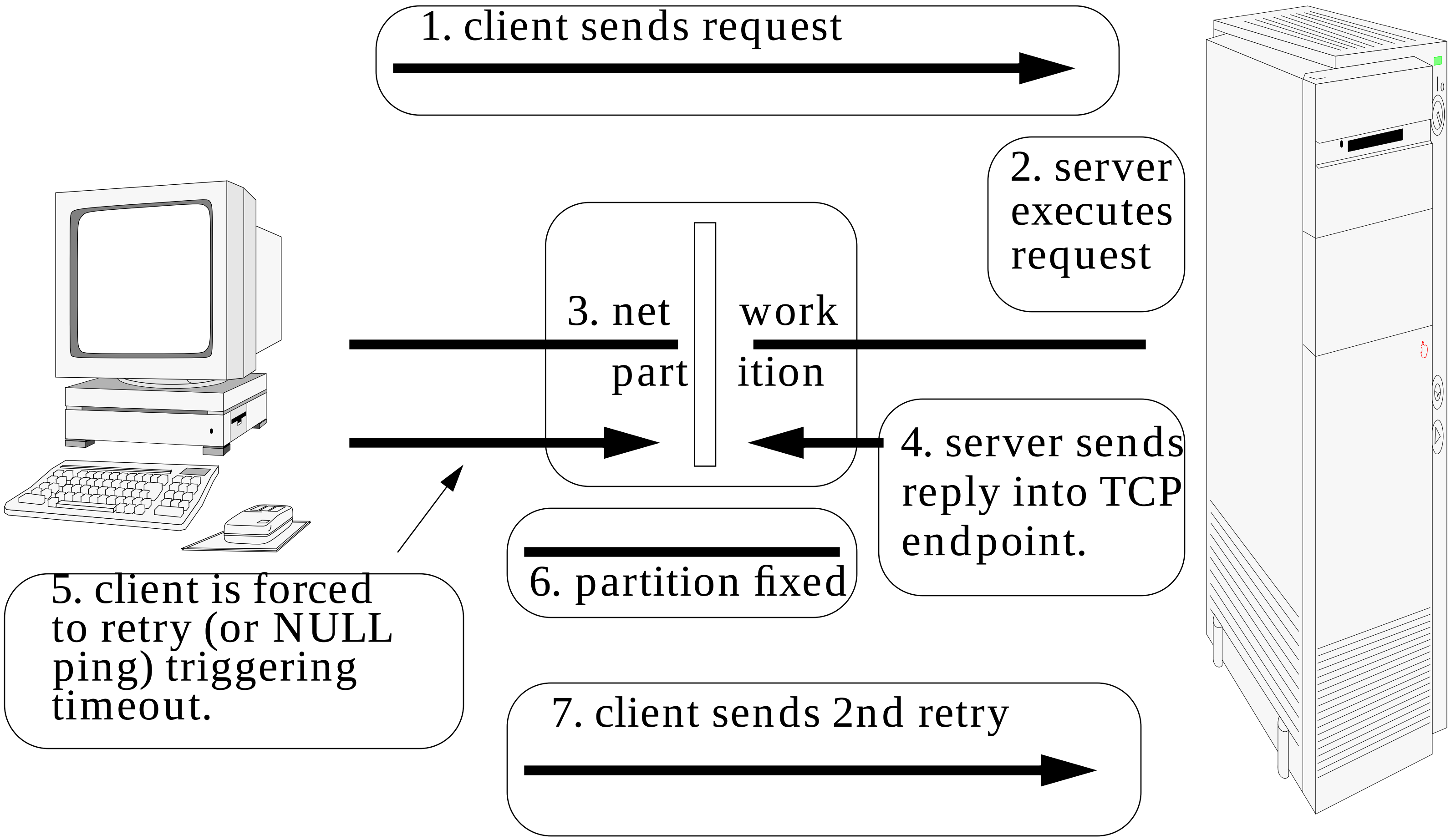
- Connections are accepted by user-level nfsd daemon using TLI.
- nfsd uses private system call:

```
ret = nfs_svc(tli_fd, “tcp”, addrmask,
          thread_count);
```

- Disconnects are also fielded by nfsd.
- New “-c #_conns” option to nfsd.

DUPLICATE REQUEST CACHE

- **Necessary to deal with partition case:**



1. client sends request

2. server executes request

3. net work part ition

4. server sends reply into TCP endpoint.

5. client is forced to retry (or NULL ping) triggering timeout.

6. partition fixed

7. client sends 2nd retry

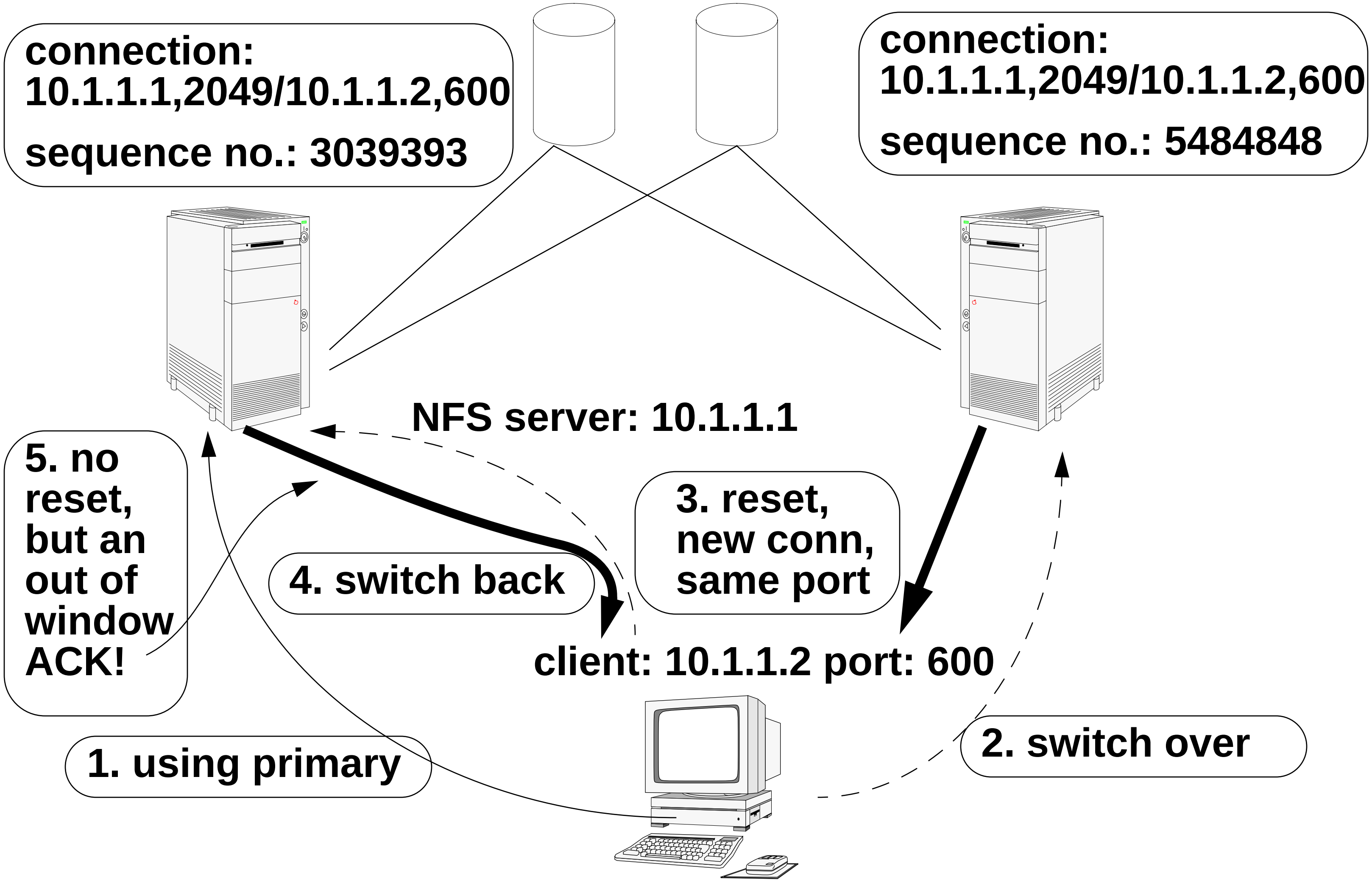
DUPLICATE REQUEST CACHE

- **Client must be careful to use the same source address (IP addresses and port) on retries after reconnects.**
    - TLI/TPI actually makes this easier than sockets do.

WAR STORIES

- **Noticed that initially, not all segments on 10-baseT were expected 1460 bytes.**
- **Didn’t notice that we forgot to increase default timeout from 1.1 secs (raised to 10 seconds now).**
- **HA-NFS product’s failover got burned by connection caching on client and server.**

HA-NFS IMPLEMENTATION WAR STORY



connection:
10.1.1.1,2049/10.1.1.2,600
sequence no.: 3039393

connection:
10.1.1.1,2049/10.1.1.2,600
sequence no.: 5484848

NFS server: 10.1.1.1

5. no reset, but an out of window ACK!

4. switch back

3. reset, new conn, same port

client: 10.1.1.2 port: 600

1. using primary

2. switch over

FUTURE WORK

- **Performance: why somewhat slower than NFS/UDP?**
  - Maybe LADDIS V3 should use TCP in the work load?
- **Default timeout of 10 seconds should be analyzed, given that the nominal RPC/UDP timeout of 1.1 seconds is over a minute after retries and backoff.**
- **Look at changing NFS client to probe unresponsive connections with NULL pings before doing a retry.**
- **Can we optimize new per NFS access checking code to once per connection?**