Clustering Samba: Problems, Pitfalls and Possibilities

By Jeremy Allison

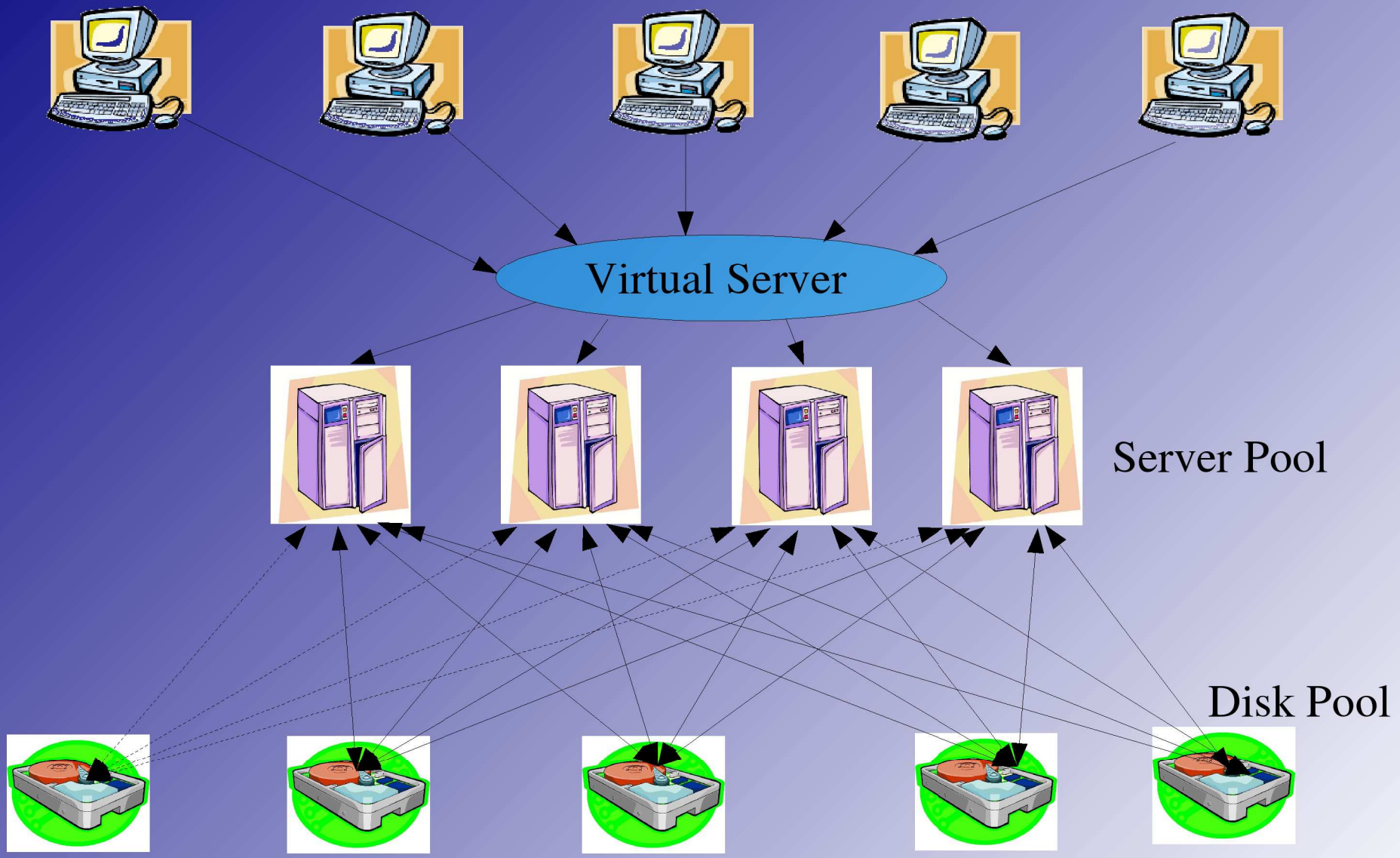
jra@samba.org

“Everyone talks about the weather, but no one does anything about it.”: Mark Twain

The ultimate goal...

- A clustered file server ideally has the following properties :
  - All clients can connect to any server.
  - A server can fail and clients are transparently reconnected to another server.
  - All servers can serve out the same set of files.
  - All file changes are immediately seen on all servers.
    - Distributed filesystem.
  - Ability to scale by adding more servers/disk backend.
  - Appears as a single large system.

The ultimate goal.



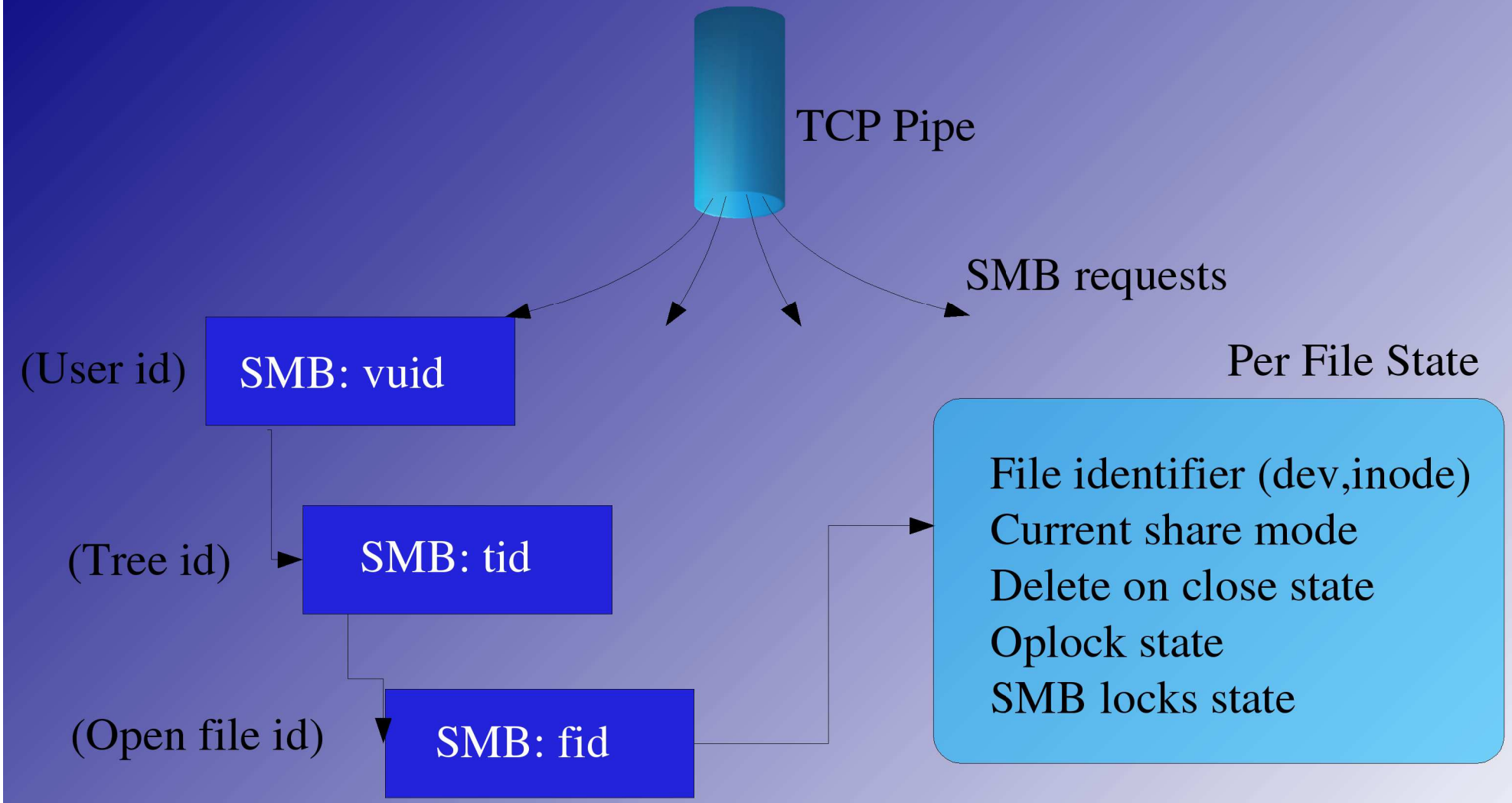
Why is this hard ?

- In a word – STATE !
  - Both TCP state and the SMB state above it.
- SMB uses TCP connections, so active failover is not seriously considered.
  - All current SMB clusters are failover solutions, they rely on the clients to reconnect.
- Servers must keep state about client connections.
  - SMB keeps a lot of state.
  - Every file open must be compared with other opens to check share modes.

Try starting from the front...

- To appear as a fileserver with one name and address, the incoming TCP streams from clients must be de-multiplexed at the SMB level and fed to different servers in the pool.
  - First decision is to split all IPC$ connections and RPC calls to one server to handle printing and user lookup.
    - RPC Printing handles are shared between different IPC$ sessions, very hard to split between servers.
  - All other servers simply provide file service.
    - Simpler problem to concentrate on.

Distributed SMB state for a file.



De-multiplexing SMB requests.

- To de-multiplex SMB requests, knowlegde of the current VUID state must be held within the front-facing “virtual” server.
  - WinXP and above have changed semantics so that the vuid, tid and fid must match for a successful operation (makes things simpler).
  - SMB requests are sent by vuid to their associated server. This code doesn't exist yet (similar to Windows 2000 Terminal Server problem)
  - May be simpler to start by exposing the server pool to the clients directly – saves the demultiplex step.

Now examine the back – the distributed filesystem.

- Many distributed filesystems exist for UNIX/Linux.
  - Many of them can be adopted for the backend purpose, so long as awareness of SMB semantics is kept in mind (share modes, locking and oplock issues in particular).
  - Common free software ones are :
    - NFS, AFS, OpenGFS, Lustre
  - The server pool can use any distributed filesystem backend if all SMB semantics are performed within this pool.

Distributed filesystem restrictions.

- If the server pool only serves SMB, oplocks may be handled within the pool without backend filesystem support.
- If the server pool also serves NFS or other file protocols then their implementations must become oplock aware and interoperate with Samba.
  - Otherwise, no oplocks – loss of performance by Windows clients.
- Protocol state must be shared across the pool.

Communication in the server pool.

- As most backend filesystems usually support POSIX semantics, it is hard to push the SMB state back into the filesystem.
  - eg. POSIX locks are signed and have different semantics to SMB locks.
- Clearly, smbd processes in the server pool must communicate quickly.
  - Non clustered Samba uses tdbs and a local loopback UDP protocol to pass this state info.
  - Clustered smbd's must use something else.

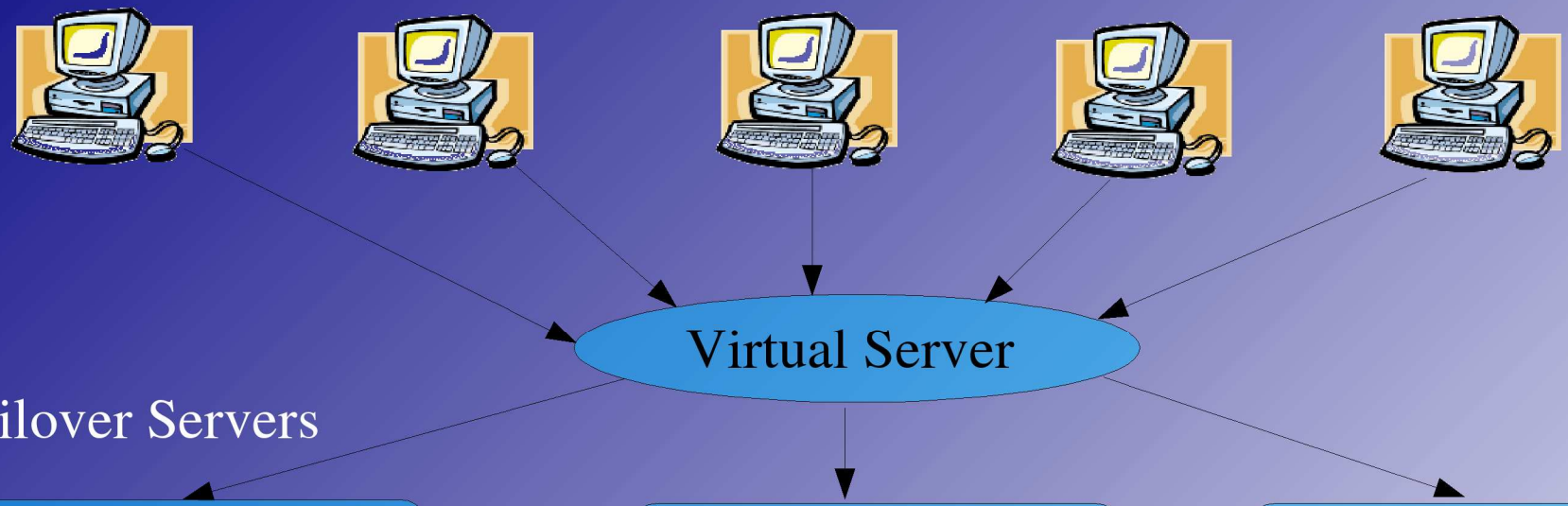
Fast communication in the server pool.

- Using filesystem shared tdb's and distributed fcntl calls is (as tridge puts it) a w%&k solution. ☺
- A new fast interconnect must be used.
  - Possibilities are a proprietary shared memory bus such as Myrinet or SCI (Scalable Coherent Interface) -which are very expensive.
  - However, Gigabit ethernet cards are now $60, gigabit switches are $150.....
  - Bypassing TCP or UDP and using raw ethernet framing may be needed to get the interconnect speed.
  - I have no data on the speed needed for this to work.

Samba modifications needed.

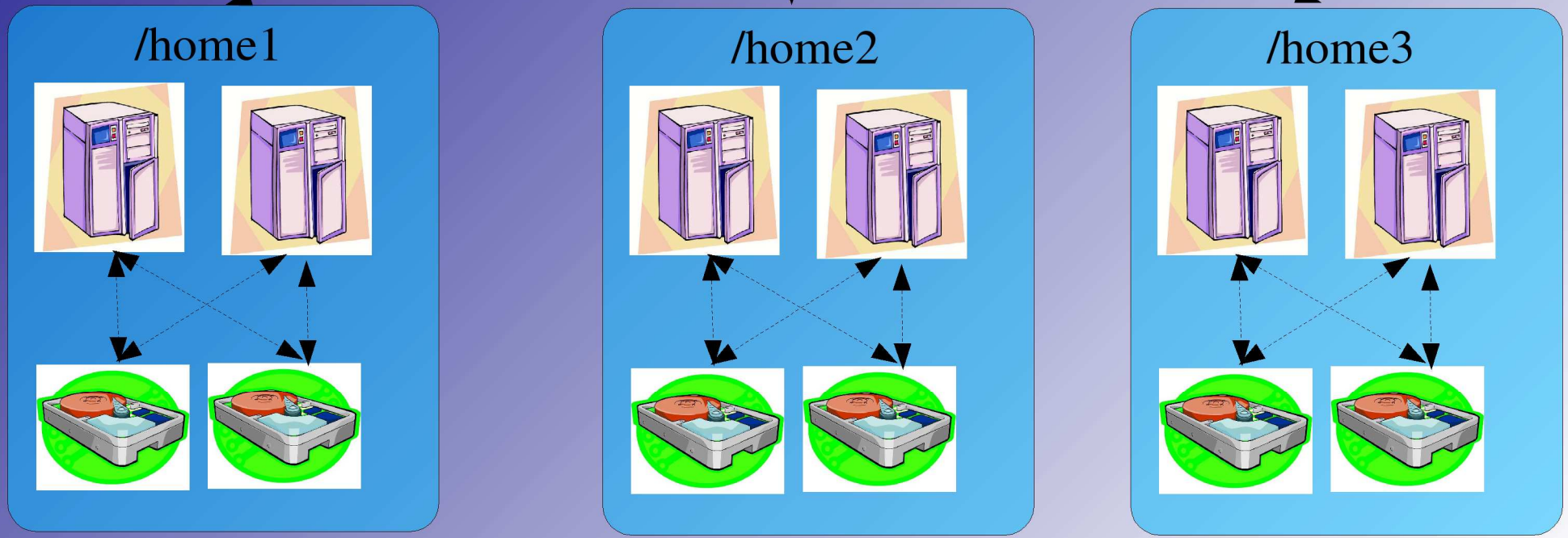
- Clustered Samba in the previous configuration needs to be modified to integrate with the fast interconnect.
  - The operations on the locking database, the share mode database and the oplock notification code must all be modified to communicate over the fast interconnect.
  - Failure semantics must be defined. Samba behaves the same way as Windows when oplock messages fail (allows the open regardless) but what about inter-machine messages ?
  - Do we use point to point (lock manager) or multicast?

How can we ease the new code complexity ?



Virtual Server

Failover Servers



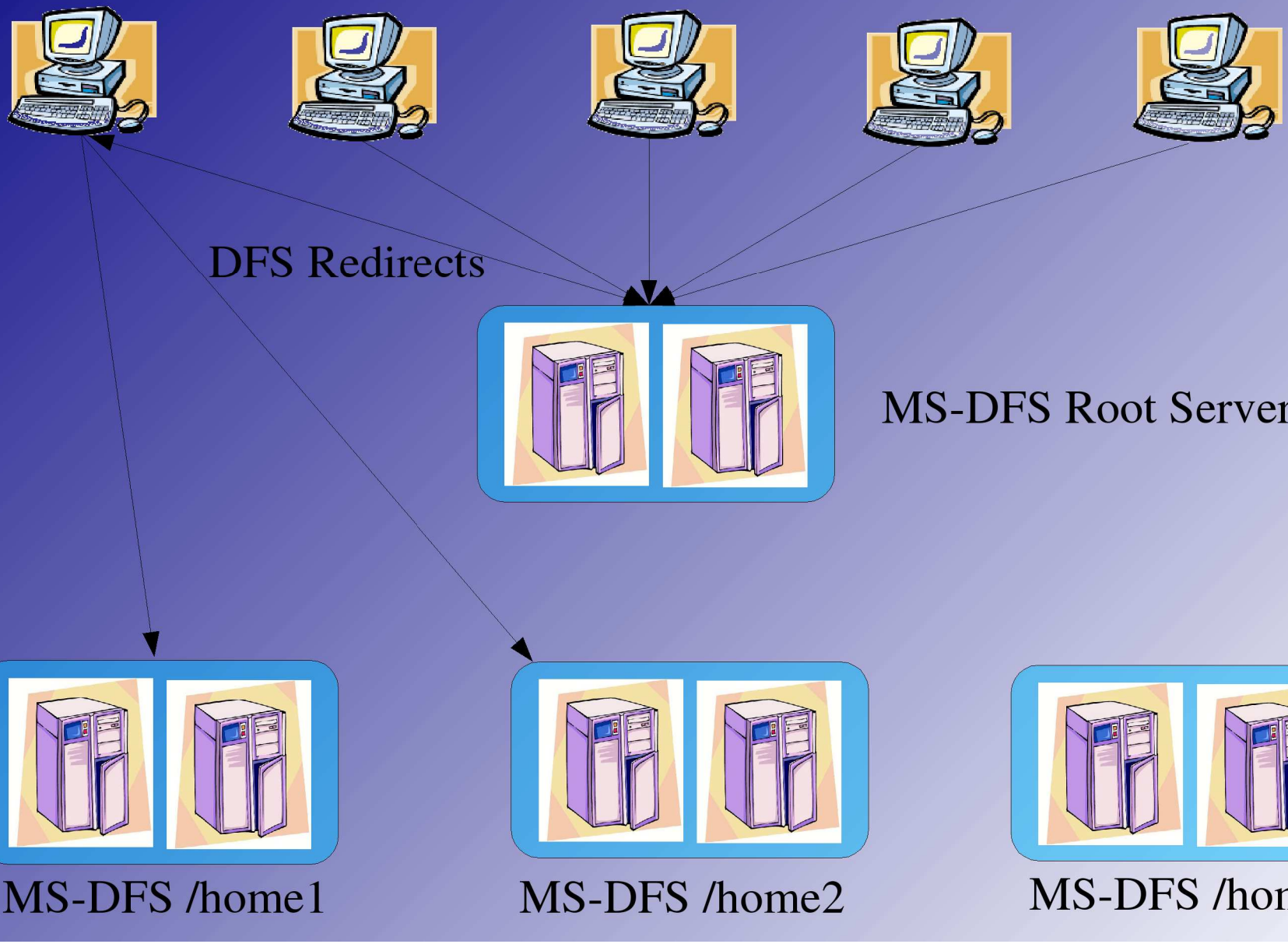
A Simpler Solution

- Allowing failover servers to handle different areas of the exported filesystem removes the problem of writing the distributed locking protocol.
  - Only one server active in a pair, no fast interconnect needed, existing high availability solutions can be used out of the box.
  - Disadvantage is management of the file namespace.
    - No longer a single coherent namespace, admins must remember what is where.
  - Frontend 'virtual server' still needed to redirect to backend servers and must have namespace knowledge kept coherent.

High availability server products.

- The failover servers must communicate in order to do the resource failover needed for standard high availability work.
  - The heartbeat signal is normally done over a shared LAN or serial interface.
- Red Hat Cluster Manager and Microsoft Wolfpack can use a shared SCSI or Fibre Channel disk partition for communication.
  - The remaining complexity is in the frontend server.
  - Maybe we can find a way for clients to handle this....

But this is starting to look like....
MS-DFS !



MS-DFS: The “poor mans” cluster.

- MS-DFS links can be used to redirect clients onto disparate backend servers.
  - Pushes the complexity into the client code – already included by Microsoft.
  - Creates the illusion of a simple, continuous namespace.
  - Even works at the file level !
- At the cost of complexity of management, a distributed (clustered ?) Samba can be created with existing Samba functionality.

Conclusions.

- Clustering SMB is hard.
  - Client failover is the best we can do.
- Most promising Samba code changes are in distributed “open files” code.
  - In conjunction with Linux single system image code could eventually create an “out of the box” SMB cluster (productised by vendors).
- Until this code is created an organisation can create the illusion using MS-DFS if they can cope with the management complexity.

References

- OpenGFS web site:
  - opengfs.sourceforge.net
- Lustre web site :
  - www.lustre.org
- OpenAFS web site :
  - www.openafs.org
- Microsoft DFS guide :

  http://www.microsoft.com/NTServer/nts/downloads/winfeatures/NTSDistrFile/AdminGuide.asp

Questions ?