Designing an NFS-based Mobile Distributed File System for Ephemeral Sharing in Proximity Networks

Nikolaos Michalakis
Computer Science Department
New York University, New York, NY

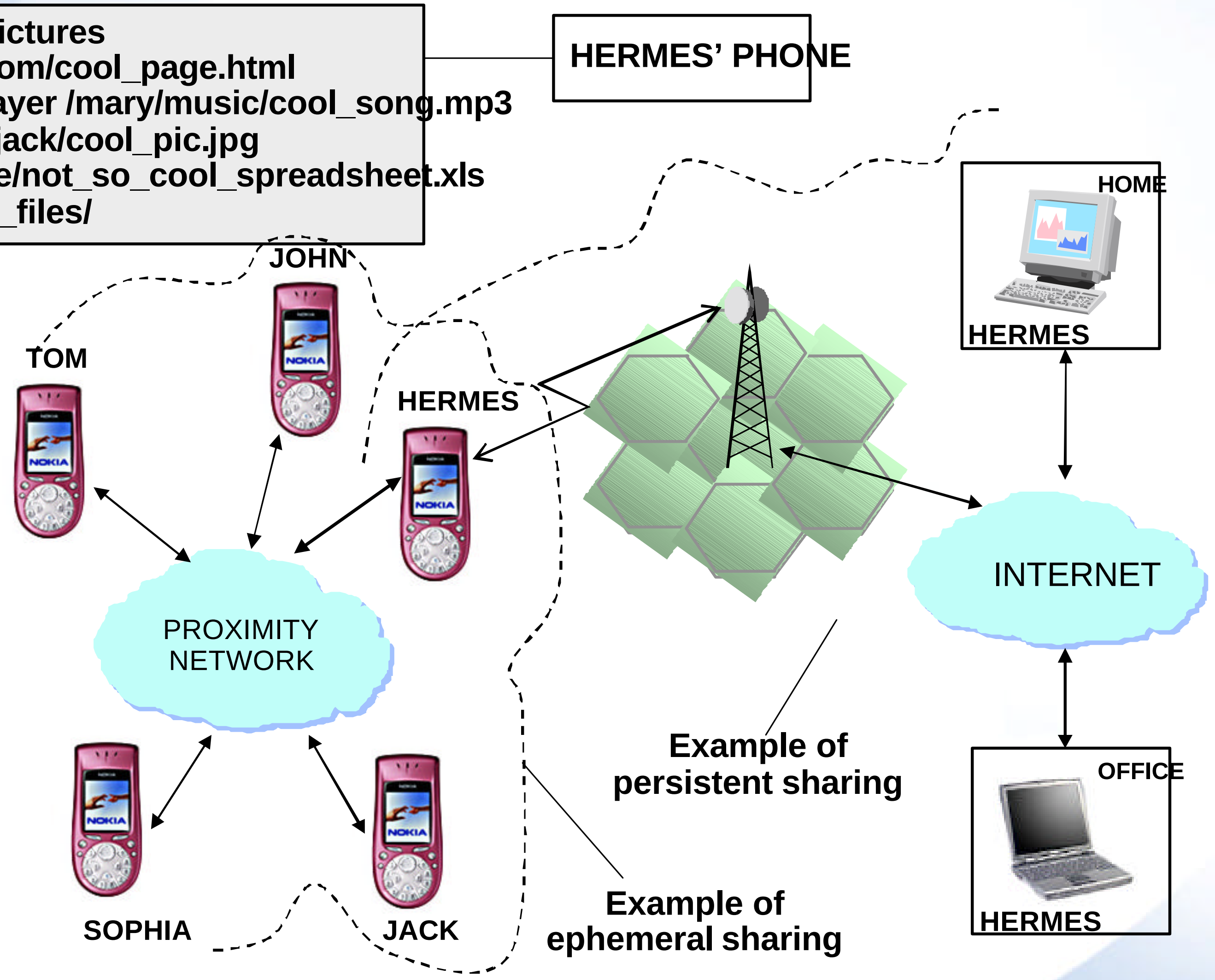
Dimitris Kalofonos
Pervasive Computing Group (PCG)
Nokia Research Center, Boston, MA

Outline

- Ephemeral vs. Persistent file sharing
- Motivation for using NFS as a basis for Ephemeral M-DFS
- Related Approaches
- NFS Performance in Proximity Networks
  - Problems with Mobility
  - Experimental and Simulation Setup
  - Performance Results
- Designing an M-DFS for Ephemeral Sharing
  - M-DFS Design Topics and Suggestions
  - An Implementation Proposal
- Future Directions

Ephemeral vs. Persistent File Sharing

```
>> cd /sophia/pictures
>> ./netscape /tom/cool_page.html
>> /sophia/mplayer /mary/music/cool_song.mp3
>> /tom/xview /jack/cool_pic.jpg
>> ./excel /office/not_so_cool_spreadsheet.xls
>> ls /home/my_files/
```



Example of persistent sharing

Example of ephemeral sharing

Ephemeral vs. Persistent File Sharing (cont.)

Ephemeral file sharing: a remote file system is discovered and mounted to enable short-lived collaboration among users

- Intended mainly for read-only file sharing
- Client caching takes place mainly to improve performance (e.g. in low bandwidth-links). It does not guarantee disconnected operation
- Deals with short disconnections (order of seconds-minutes) due to intermittent connectivity at the link level. No state is expected to survive long (as defined by the user) disconnections
- If files are to remain accessible in the long-term, user “saves” local copies

Persistent file sharing: enables users to keep a personal distributed file repository among their devices (e.g. phones, PC’s)

- Intended mainly for read-write file sharing
- Client performs (Coda-like) hoarding when fully connected, can operate on files when disconnected, reintegration-conflict resolution when reconnected
- Disconnections can be arbitrarily long (order of hours-days)
- Files are accessible in the long-term without “saving” local copies

Why Use NFS as Basis for Ephemeral M-DFS?

- NFS perhaps the most successful DFS to date. Widely used, well tested
- Native support for NFS in Linux kernel. Appealing for Linux-based mobile devices and experimentation
- NFS interface has made it attractive as a basis to build DFS with additional features. Tools available (e.g. SFS toolkit) for user–level implementation. Makes portability across platforms possible
- Light mounting and unmounting process allows discovering exported files systems and mounting on demand
- Allows RPC-based access to remote files, without requiring the transfer of whole files
- Not as complex as other popular DFS for mobility (e.g. CODA). More suitable as a basis to build M-DFS for resource-constraint devices

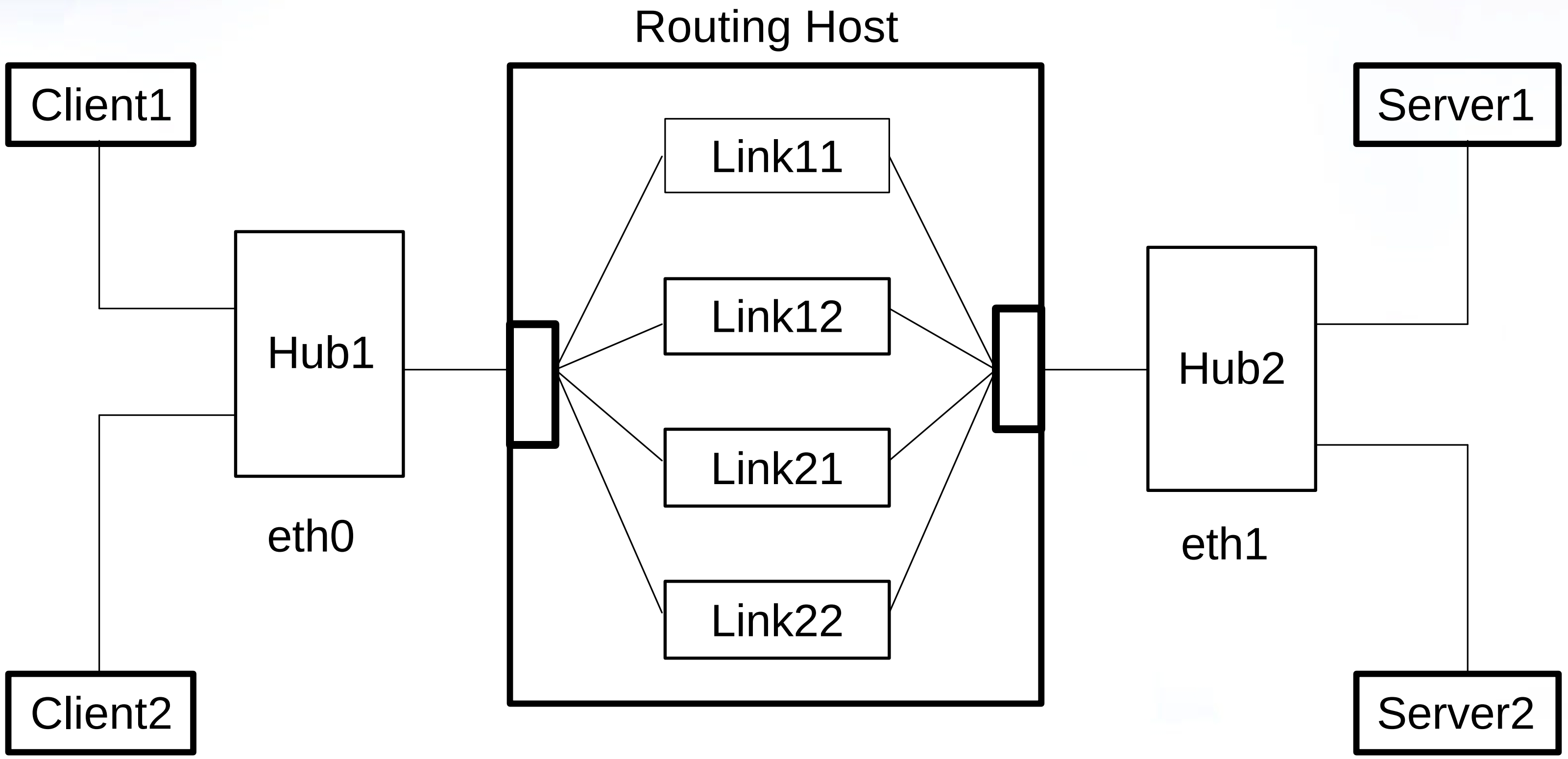
Related Approaches

- CODA/Intermezzo
  - Supports true disconnected operation
  - Client performs aggressive caching (hoarding) when connected
  - Optimizations available for weak connectivity
  - Intended for Persistent sharing
  - Some design aspects applicable to Ephemeral sharing. However, too complex to be used as is. Removing functionality tricky
- NFS/M
  - NFS modified to support (CODA-like) disconnected operation
  - Requires Kernel modifications
  - Intended for Persistent sharing. Redundant functionality for us
- SFS/LBFS
  - User-level DFS on top of NFS to add security, low BW enhancements
  - SFS toolkit available that deals with issues of loopback NFS

NFS Performance in Proximity Networks: Experimental Setup

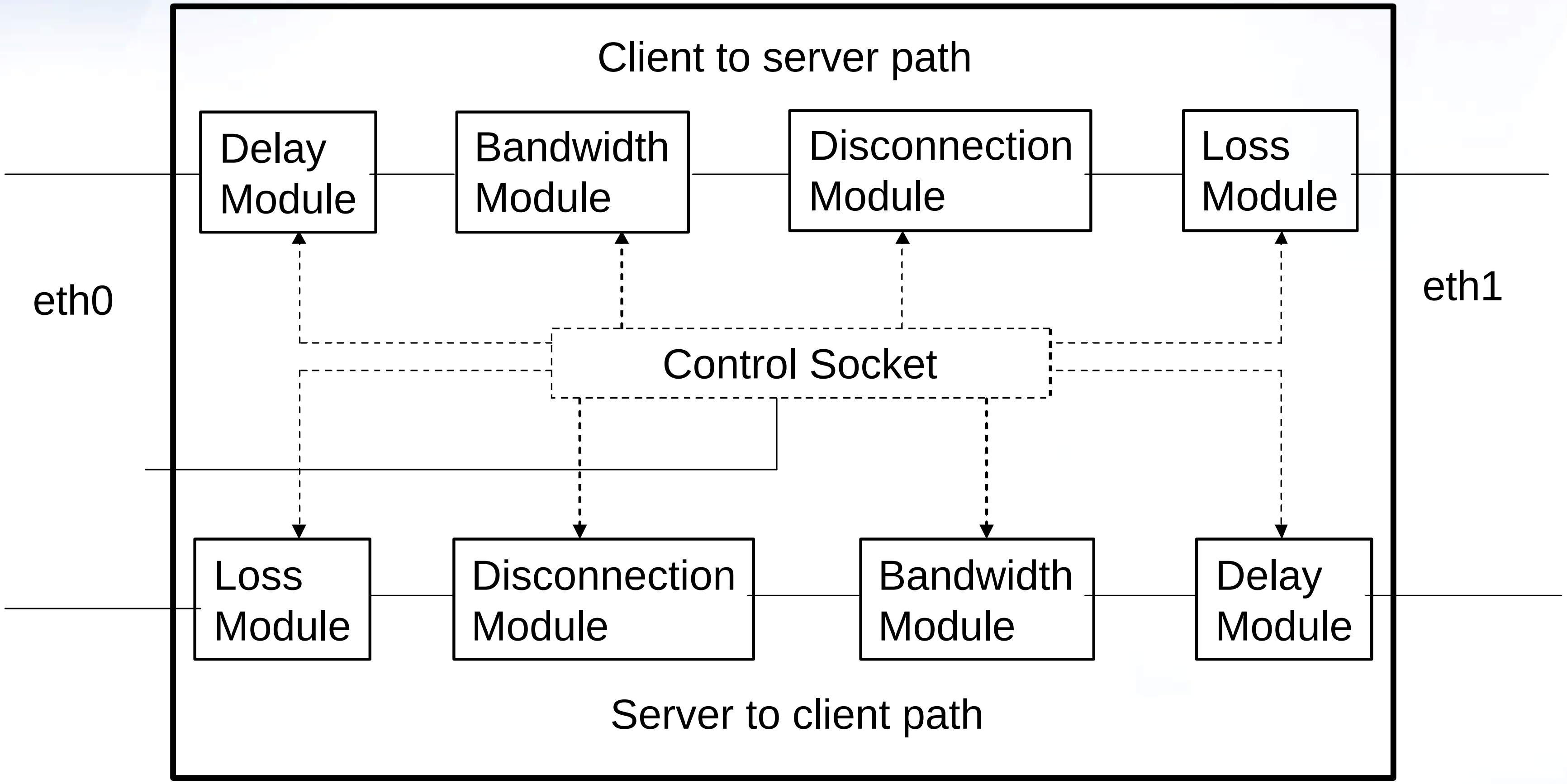
- NFS v3 client and server on two laptops running Linux 2.4.18
- Mobility: IP address changes by attaching to different AP
- Intermittent connectivity: disconnections by releasing IP address
- 802.11b measurements
  - Transfers with both client and server on same AP
  - Data rate selection: 1Mbps, 2Mbps, 11Mbps
- Bluetooth measurements
  - BlueZ stack v2.2 with PAN profile 1.0
  - Bluetooth v1.1 USB adapters
  - Simple piconet with a Master (NFS Server) and a Slave (NFS Client)
  - Increasing distance increased the RTT because of link-level re-Tx
  - Data rate selection: DH1, DH3, DH5

NFS Performance in Proximity Networks: Simulation



Network simulator using the ‘Click’ modular router

NFS Performance in Proximity Networks: Simulation (cont.)

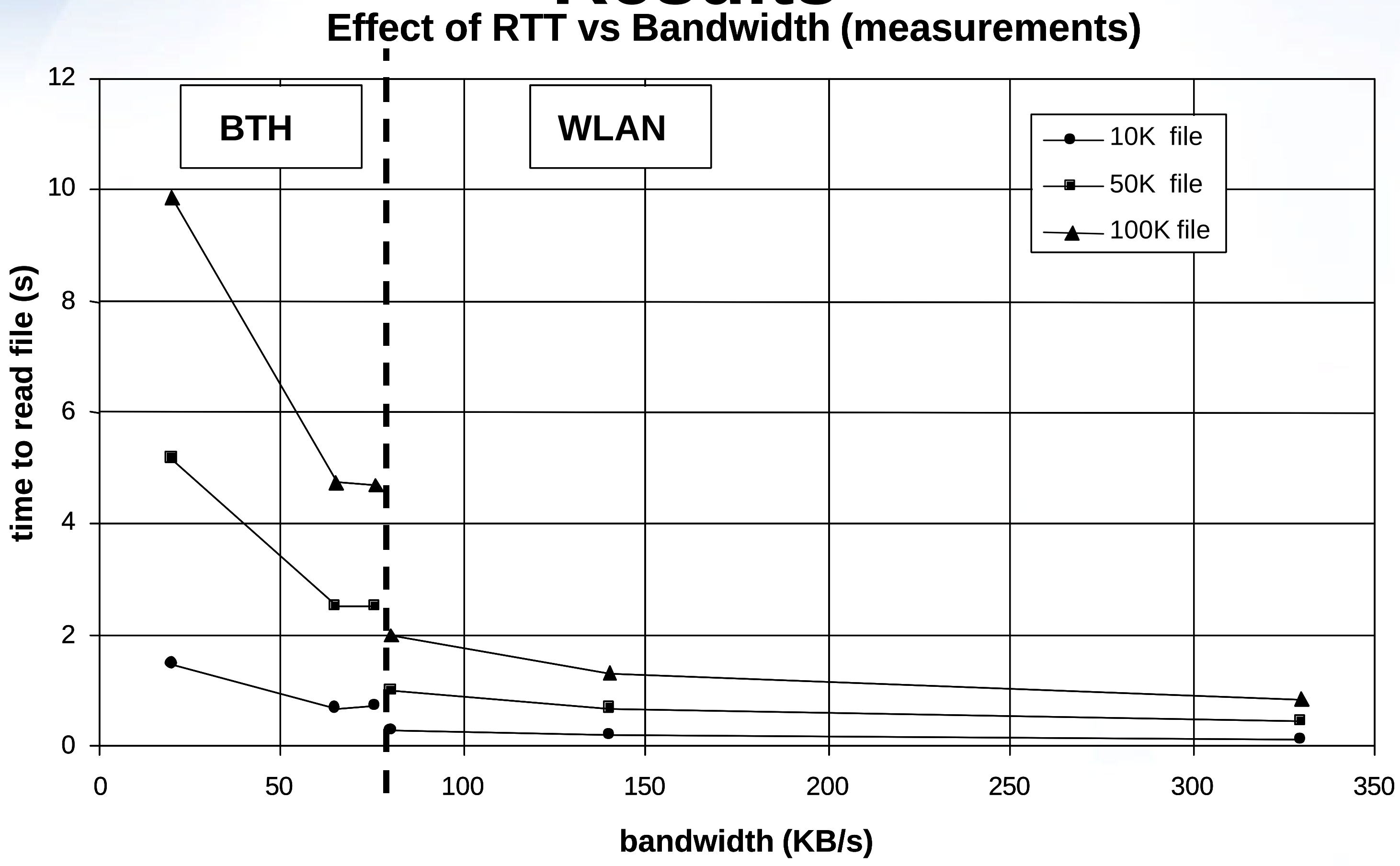


The configurable link

NFS Problems with Mobility and Disconnections

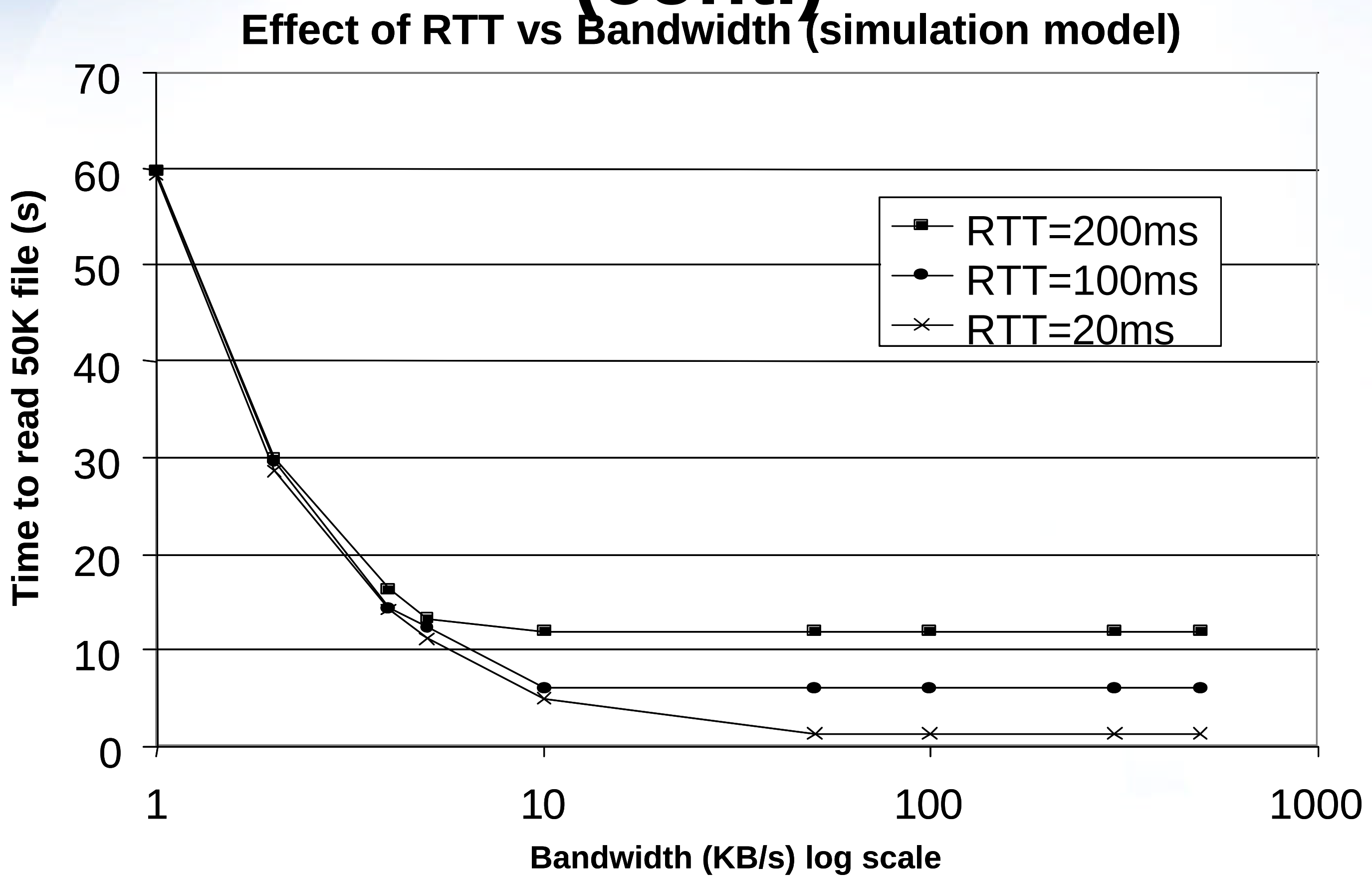
	Client has mounted	Client not mounted (server already started)
Client IP address change:	Client receives a “stale filehandle” error. The client is using a filehandle that is exported to another client address.	Client receives an “access denied” error. The server has already created a list of client addresses that can access the exported file systems.
Server IP address change:	Client is trying to contact old address. If hard mounted then it blocks forever. If soft mounted it timeouts and fails (I/O error).	No errors.
Client network interface down:	Network unreachable error, no matter whether data cached or not at the client.	The mounted directory is empty, i.e. not mounted to anything.
Server network interface down:	If hard mounted client blocks forever. If soft mounted it timeouts and fails (I/O error).	Client cannot mount. Mount process gets “can't get address for server” error.
Client disconnects and reconnects with old IP address:	As long as the reassignment does not take longer than the timeout, the client only experiences a delay.	No errors.
Server disconnects and reconnects with old IP address:	As long as the reassignment does not take longer than the timeout, the client only experiences a delay.	No errors.

NFS in Proximity Networks: Performance Results



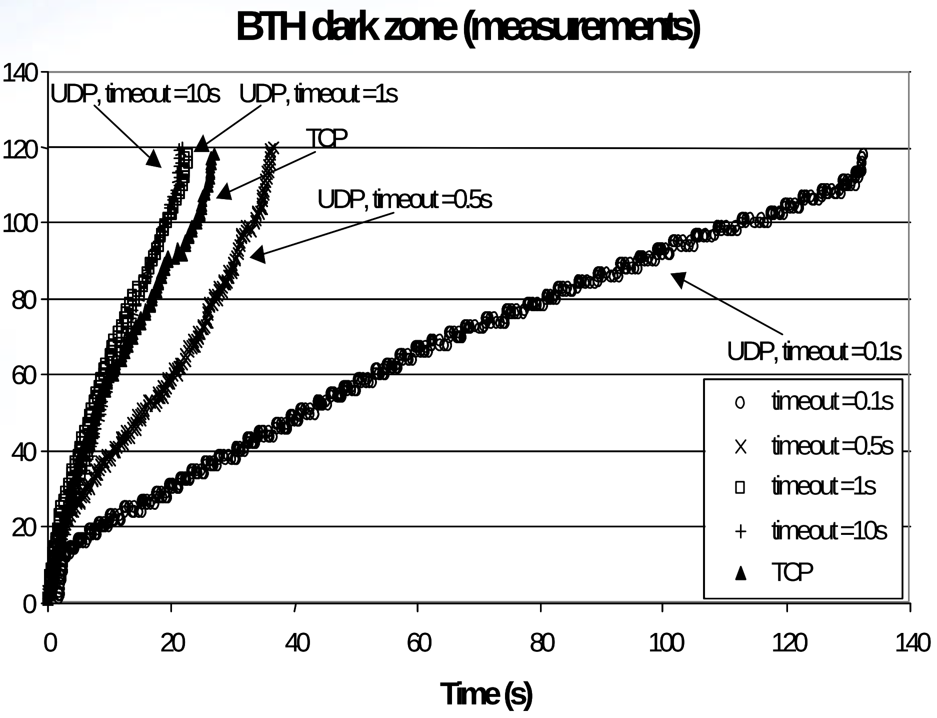
$$T \approx N \times (RTT + \frac{rpc}{bw})$$

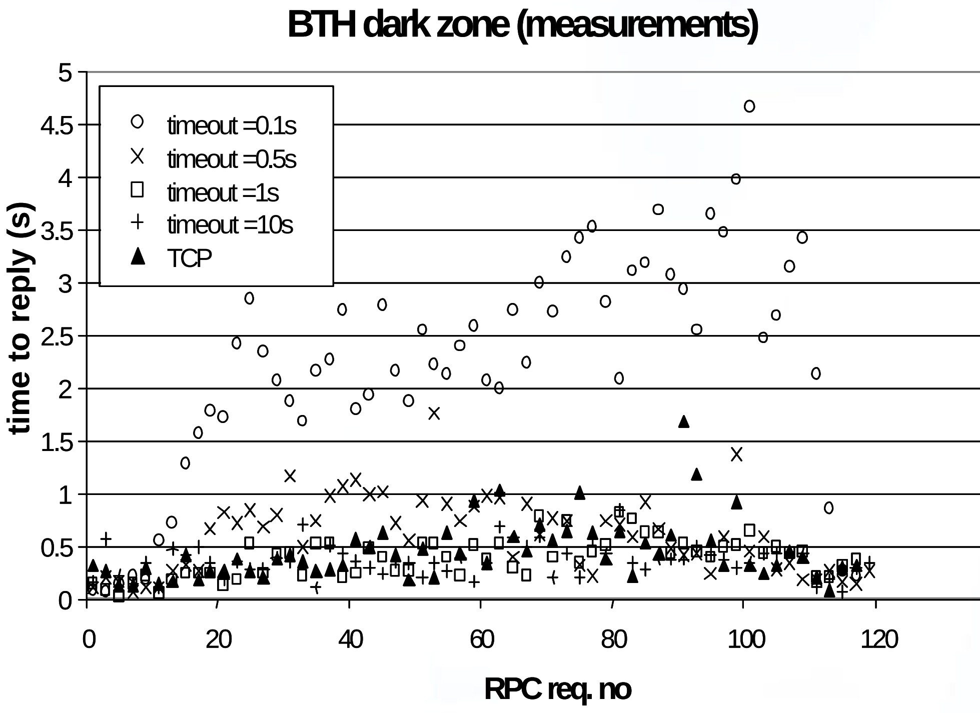
NFS in Proximity Networks: Performance Results (cont.)



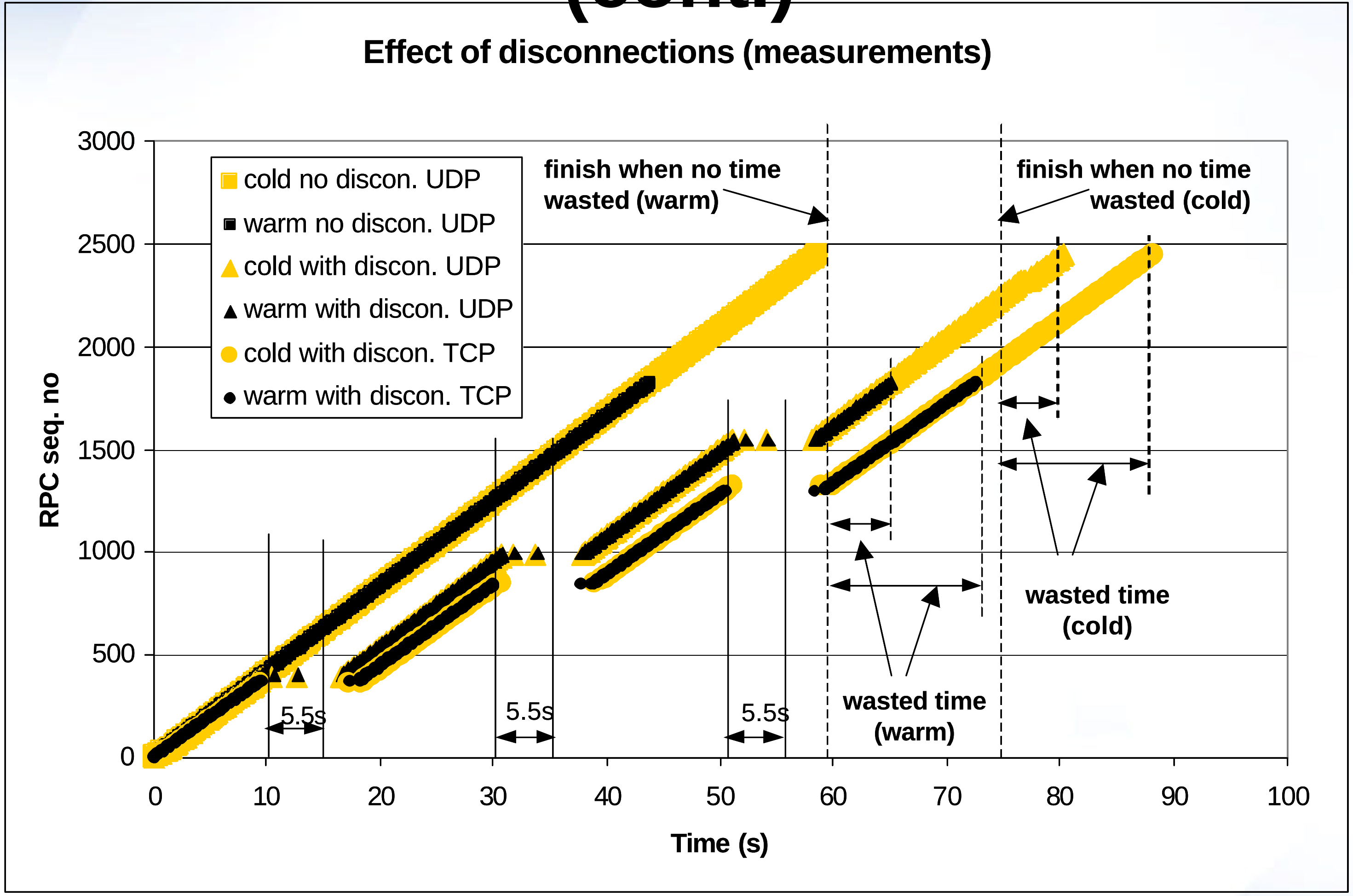
$$T \approx N \times (RTT + \frac{rpc}{bw})$$

NFS in Proximity Networks: Performance Results (cont.)

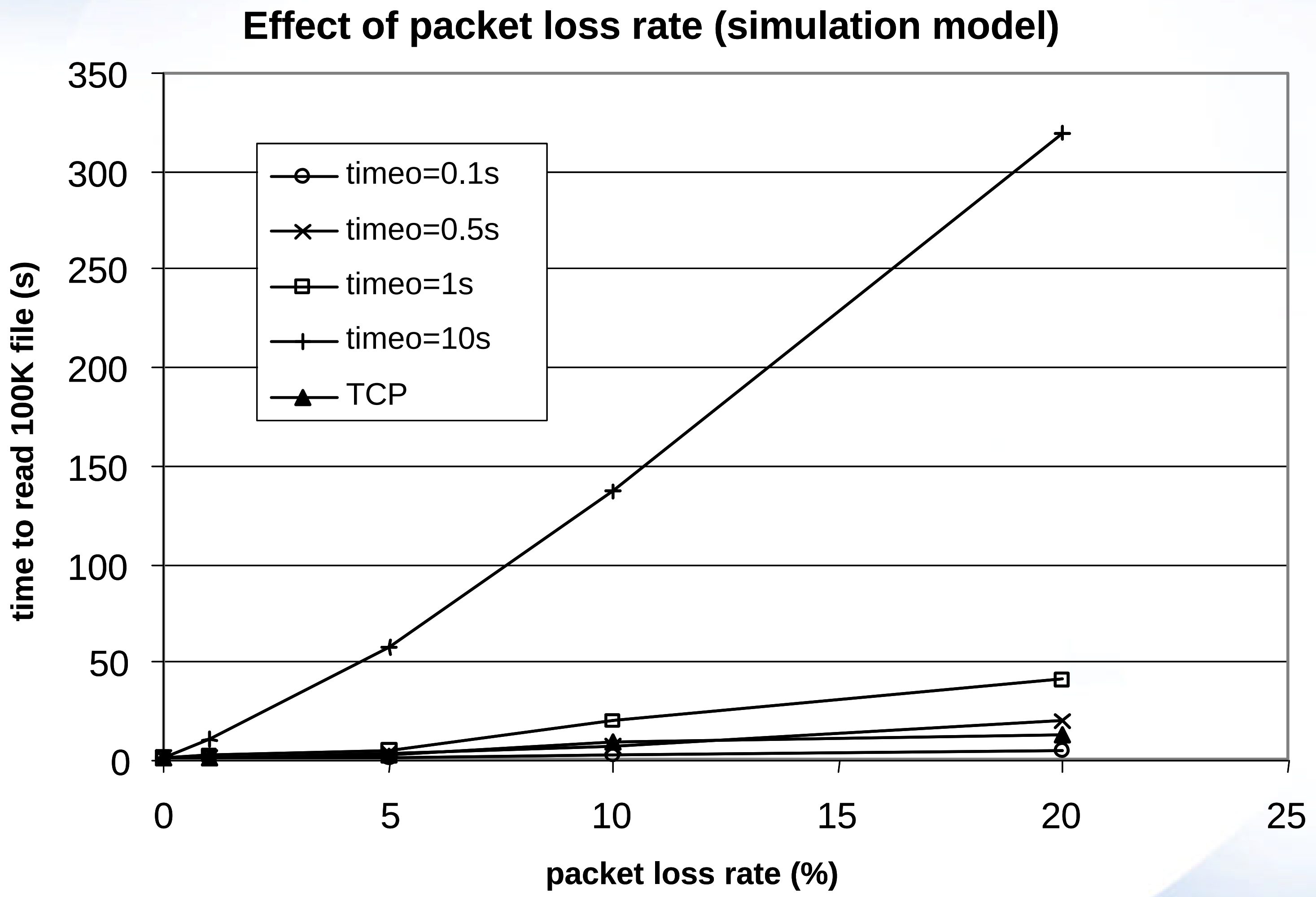




NFS in Proximity Networks: Performance Results (cont.)



NFS in Proximity Networks: Performance Results (cont.)



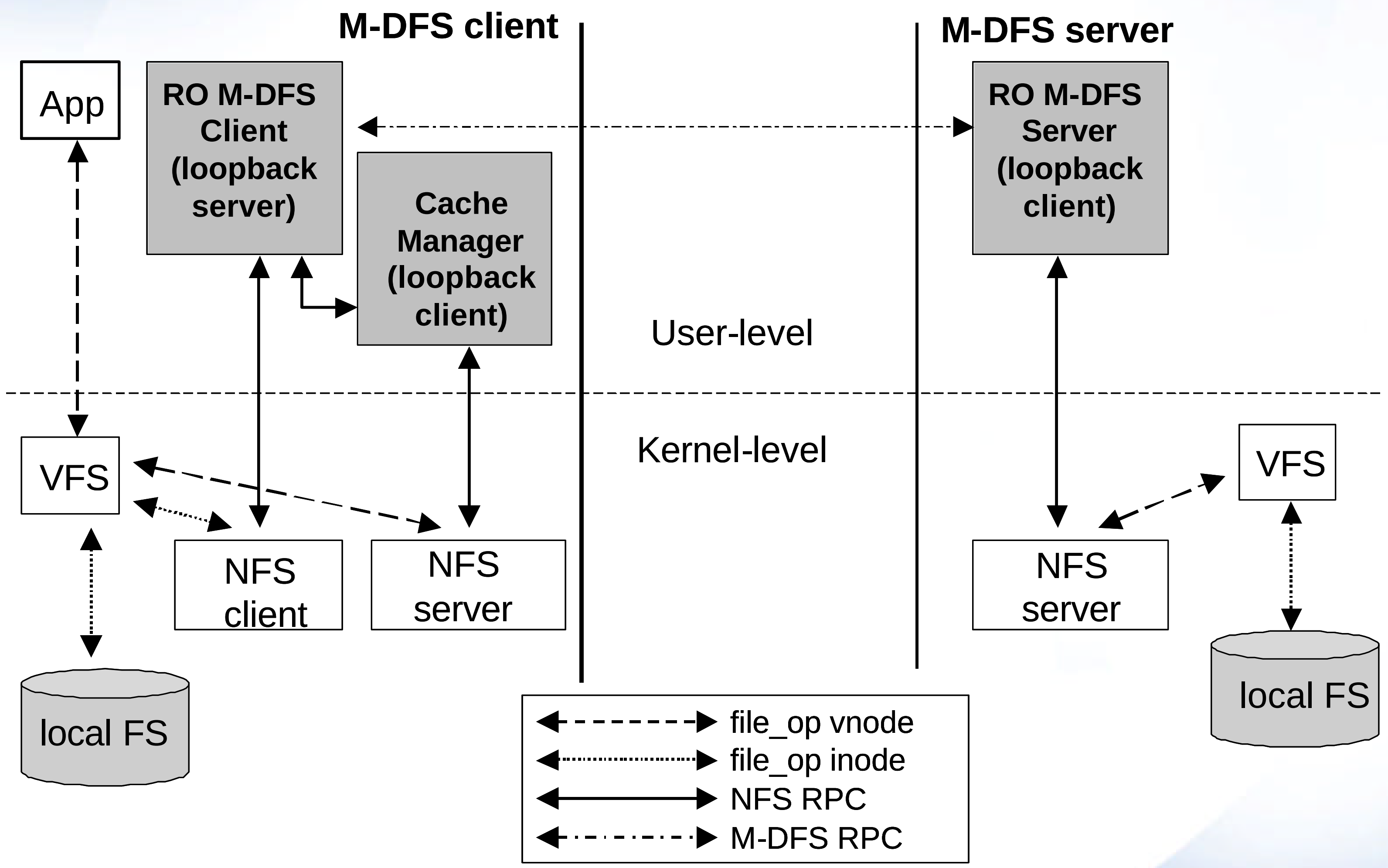
M-DFS Design Topics and Suggestions

- More aggressive caching to support temporary disconnections
  - In the case of RO ephemeral sharing caching can be simplified, e.g. callbacks with leases: close-to-open consistency, reduced traffic
  - In addition to file and directory contents, client caches metadata such as picture thumbnails or mp3 samples
- Automatic session recovery to deal with errors because of client/server mobility and loss of connectivity
  - Session support mechanism: client and server have active session (server exported and client mounted the file system), data in transit. Need to decide whether to resume or drop pending sessions upon reconnection
  - Session recovery mechanism: the server has started a session and is waiting for client to join (server only exported file system). Need to preserve the file system state related to mounting
- Techniques for traffic reduction
  - data compression techniques, application specific data reduction techniques using metadata (e.g. thumbnails, mp3 samples)

M-DFS Design Topics and Suggestions (cont.)

- Enhanced metadata to help deal with weak connectivity networks
  - Application specific metadata (e.g. thumbnails, mp3 samples) are opaque to the M-DFS. The M-DFS allocates extra space and an API for applications to use them
  - M-DFS specific metadata are seen only by the FS and the OS. Can be used to improve performance, provide enhanced capabilities and make the system more adaptive to mobile conditions. Example: estimated file transmission time
- Access control and distributed authentication
  - ACL provide better security than Unix access control
  - Use “ephemeral” ACLs, i.e. ACLs that are created and updated by the server owner on the fly. Remove stale entries in ACLs to avoid long ACLs with few active user.
  - Distributed client/server authentication an issue without ‘trusted authorities’. Simplest solution would be to use offline means, e.g. user-to-user communication
- Naming and file system removal
  - A global name space not required. Server repository may be identified explicitly by the device name that exports it. Uniform name space desirable

An Implementation Proposal



Future Directions

- Conclude the proposed implementation and experiment with the design suggestions. Assess performance of different design choices
- Experiment with the M-DFS for ephemeral sharing over ad-hoc testbed using Bluetooth
- Explore alternatives to NFS to base design of M-DFS, e.g. HTTP-based approach