Shared Name Spaces

Geoff Collyer (geoff@collyer.net)
Russ Cox (rsc@swtch.com)
Bruce Ellis (brucee@chunder.com)
Fariborz "Skip" Tavakkolian (fst@9netics.com)

ABSTRACT

One of the strengths of the Plan 9 operating system[4] is its ability to multiplex name spaces efficiently. Another key Plan 9 concept is the 9P[5] file protocol, and its uniform application to all resources – stored files and devices. The Shared Name Space system is a technique to dynamically manage name spaces exported by a group of users into an aggregate name space, and reflect it back to each group member. The net effect is a mechanism for providing a virtual network service that competes favorably with peer-to-peer networks, or central-storage file sharing and has advantages over both. Although currently the only resources that are sharable are stored files, the system allows sharing of other types of resources in the future. These would include mouse, keyboard, and display, making services such as remote desktop access or on-line collaboration and conferencing possible.

The *Rangboom™ Adhoc Virtual Networks*[1] service is a commercial endeavor based on this technology. It provides remote file access and file sharing service to broadband Internet users. It is initially targeted for devices running Microsoft Windows, Linux and Mac OS X.

1. Preface

The problem that Shared Name Space addresses is how a typical user is able to securely access his/her resources or discover and use resources belonging to a group over the Internet. As the adoption of broadband access increases, as the ratio of computing devices per user increases and as the Internet users become more savvy, there is an increasing need for and expectation of ubiquitous access to all resources on all personal electronics, from any other. It is interesting to note that many of these trends had been predicted[6]. From the beginning the goal has been to address real user requirements. Popularity of peer-to-peer or central-storage sharing services provided further evidence of the necessity of a secure and selective file sharing service.

A majority of the users are expected to be Windows, Linux and Mac users; the client side effort is consequently heavily focused on Windows, Mac and Linux environments. The system has had one major revision, the most significant of which was the invention of the name space manager.

2. Overview

The Shared Name Space system allows a group of users with high-speed Internet connections, to export parts of their computer's resources – primarily stored files – to other members. A user uses a client – the Shared Name Space agent – to export his resources and import those of his group. The agents and the server speak 9P.

The system has several components not all of which are needed to provide the basic functionality of combining name spaces and cross-connecting them. Only the components directly essential to the topic are covered in detail.

The functions of the system are spread across several components; they are:

- **User Agent** is a *drawterm(8)* [2] derivative that has been augmented to include a name

[1] https://www.9netics.net
[2] Originally written by Sean Quinlan

space import function and user interface for configuration. The terms User Agent and Client are synonymous.

- **Agent File System** is an optional component of the User Agent. It is a translator that represents the imported 9P file system as a file system in the client's native OS. For example in Windows this is an Installable File System driver and in Linux is a component similar to *v9fs*[3].
- **Name Space Manager**, affectionately called the *janitor*, manages all group virtualizations and group name space management. Other helper utilities that work with *janitor* are the *reflect* utility that reflects the aggregate group name space to each client and modified versions of *exportfs* and *srvfs*.
- **Administrative System** is a set of file servers and components that automate account creation, and membership management of Shared Name Spaces.

The sequence of operations can be summarized as follows:

1. An agent contacts the server and invokes snsd.
2. The user is authenticated and snsd posts the agent's exported name space to /srv and invokes the name space reflector utility.
3. The reflector informs the name space manager of the new user's arrival to a group and serves the group's name space for that user back to the agent.
4. the name space manager adds the user's name space to it's group's name space.

3. User Agent

In order to make the system as simple as possible for its target audience, several alternatives were considered. The ease-of-use requirement meant that the system had to function without requiring the user to have any specific knowledge of IP addresses, firewall configurations or security settings. Further, the user should not be forced to learn a new tool or paradigm. As much as possible the user should be able to deal with shared resources the same as local resources.

Many of these requirements were already addressed in *drawterm*. Drawterm is essentially a file server which exports a number of resources – including local files, mouse, keyboard and display – to a cpu server. It supports the shared key authentication and encryption. What was missing were the graphical user interface and the mechanism for importing the shared name space.

The most natural representation of the shared name space is a file system; This means that, for each user environment, the shared name space should be represented as a native file system.

One of the greatest challenges has been matching 9P to native file system protocol. Windows, more than other desktop OS presents a challenge here.

4. Miscellaneous Servers

An administrative server was built to let the users create and manage their accounts via a "Web" interface. The http server provides a gateway to these administrative functions.

Users' expectation that some level of file access functionality be available through Web browsers dictated the requirement for a Web user interface. Adding a Web interface to the shared name space server has been easy and the gateway functions are relatively simple.

5. SNS Server

This component is perhaps the most interesting for Plan 9 developers. The first attempt was a straight forward implementation of a reasonable design based on evolving requirements. The design changed as requirements became sharper and we experienced the system in typical usage.

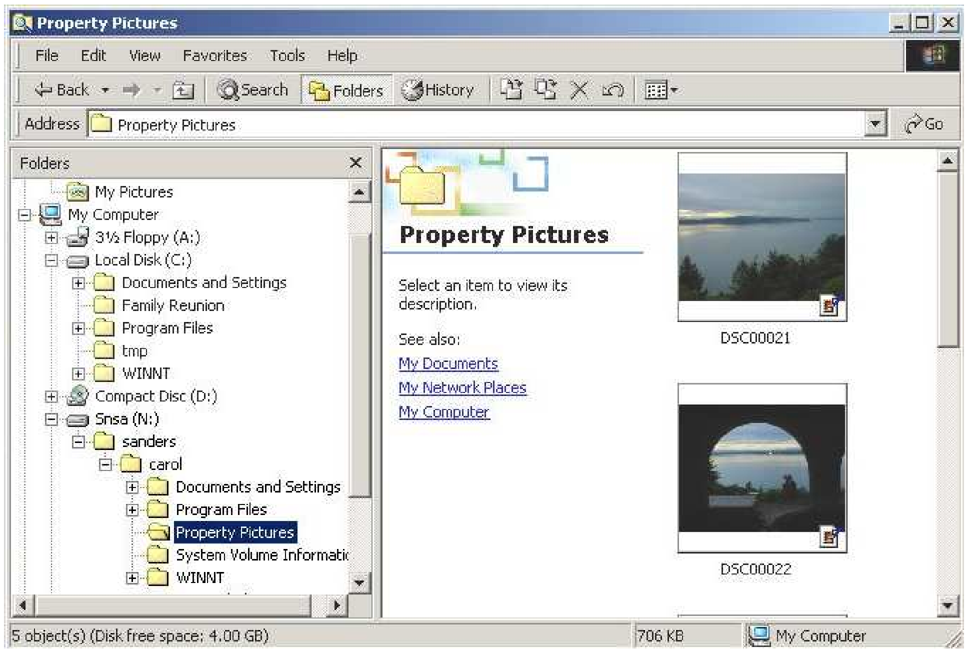


Figure 1: User's Perspective – Windows IFS

5.1. Take 1: Muxfs

The first attempt at a name space manager was to implement a filesystem. The file system, known as *muxfs* began as a modification of *exportfs(8)*. It provided the basic functions that provide a name space switching fabric and caching.

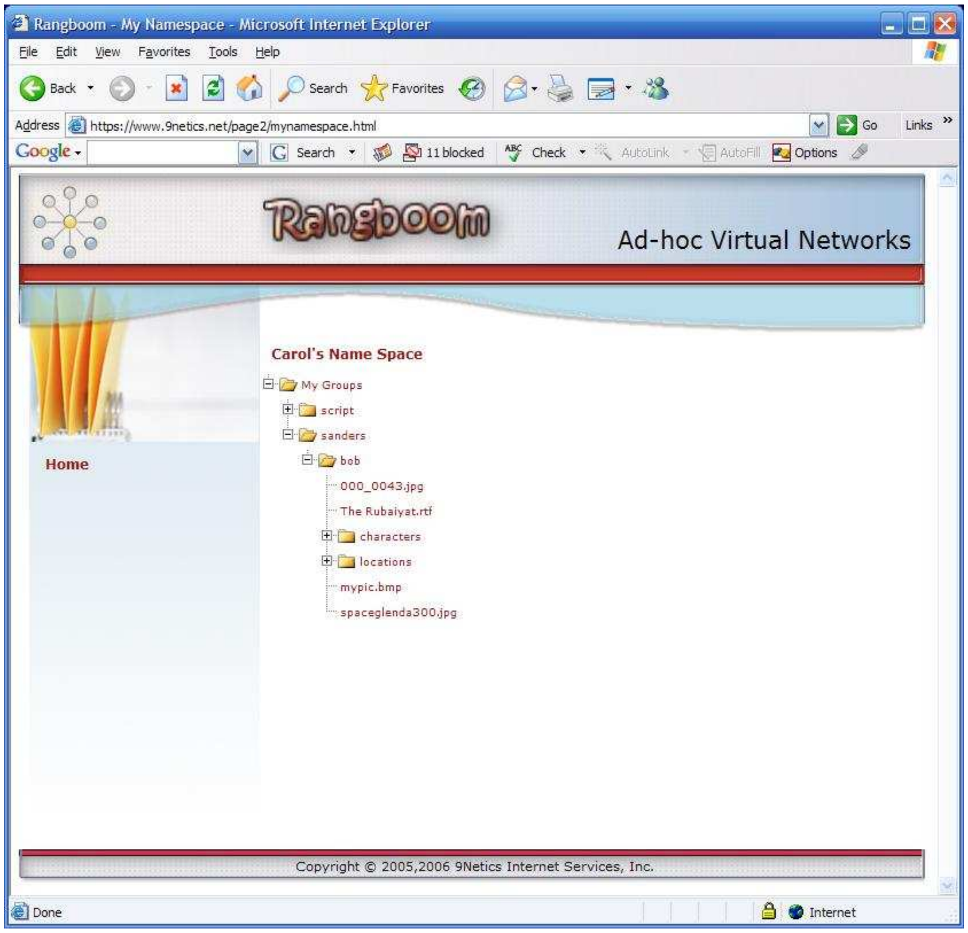
The cache was designed in to cope with anticipated device file sharing. If multiple agents wanted to read the same device file, – for example `/dev/draw` or `/dev/screen` – *muxfs* would provide a the fanout through its cache.

For each collection of agents belonging to a given group, a single muxfs was required. The muxfs was started by the first user into the group, and would stay around until all agents disconnected. Subsequent users were handled by the namespace reflector *reflect*. The process was (roughly):

```
srvfs sns.$group.$user /mnt/term
if (! test -e /srv/sns.$group) {
    muxfs $group /mnt/sns/$group
}
if not {
    mount /srv/mux.$group /mnt/sns/$group
}
exportfs /mnt/mux.$group $clientfd
```

This approach lacked performance. A thorough analysis[1] showed that each request would involved four transits through *devmnt*. The following are number of handoffs that take place on the server to deal with a request originated by `bob` to walk to some part of `carol`'s name space, before `carol`'s client receives the request.

1. *devmnt* gets Rread from `bob` containing a T message – e.g. Twalk to /n/mux/sanders/carol
2. *exportfs* corresponding to `bob`'s client gets the Tmesg
3. *devmnt* receives the Twalk for /n/mux - a pipe


Rangboom - My Namespace - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address https://www.9netics.net/page2/mynamespace.html
Go Links
Google Search 11 blocked Check AutoLink AutoFill Options
Rangboom
Ad-hoc Virtual Networks
Home
Done
Internet

Carol's Name Space

```
My Groups
  script
  sanders
    bob
      000_0043.jpg
      The Rubaiyat.rtf
      characters
      locations
      mypic.bmp
      spaceglenda300.jpg
```

Copyright © 2005,2006 9Netics Internet Services, Inc.

Figure 2: User's Perspective – Dynamic HTML View

4. *devpipe* handles the walk, to *muxfs*
5. *muxfs* reads the Twalk, starts a slave to handle the walk
6. *devmnt* makes a new T message and sends it to `carol`'s fd …
7. *devpipe* … which is a pipe …
8. *srvfs* … to `carol`'s /mnt/term …
9. *devmnt* … which is a mounted channel; creates and sends a new T message …

The analysis also included a set of recommendations. It was clear we needed to take advantage of the name space multiplexing in the kernel. At the same time, we were questioning and reconsidering the requirement to implement caching. These became the basis for a better design.

5.2. Take 2: Janitor

A key function of the Plan 9 kernel, is the multiplexing of name spaces. So it was natural that as the requirements were focused and as the essential functions were being understood, we hit on the idea of a name space manager. The revelation greatly simplified the component. As the saying goes, *Perfection (in design) is achieved not when there is nothing more to add, but rather when there is nothing more to take away.*[3] It certainly applied here.

[3]Antoine de Saint-Exupèry

Janitor manages all Shared Name Spaces on a server. The updated *reflect* is also simplified; it performs the following:

```
srvfs sns.$group.$user /mnt/term
notify janitor $group.$user
exportfs /mnt/sns/$user/$group $clientfd
```

Janitor will add the user to the group's name space and prepare the user's name space by scanning for group members that are currently connected. Once the user's name space has been updated, *reflect* is informed to serve the shared name space to the user. *Janitor* takes care of clients that have disconnected and performs the clean up.

The system can add other name spaces into the group's or the user's name space from directories and services at the server. For example, *Janitor* can add jukebox (read-only) and storage (read-write) name spaces to a shared name space. A user with access rights for "premium content" would, for example, be able to access a music or video content. A user with access rights for "archiving" would have access to a central storage space.

6. Kernel Modifications

The system has required some changes to the kernel. The most significant effort was adding the main function of *exportfs* into the kernel. The reason to move exportfs into the kernel was to implement the Twalk handler more efficiently, using direct access to the device walk functions.

7. Future Work

Future enhancements will concentrate on tighter integration of the agent into the client operating system and enhanced graphical user interface for controlling file sharing options. Windows and Linux clients are available and a Mac OS X client is in the works. Some enhancements are required to better match the requests in the client file system to 9P.

7.1. 9P Extensions

It comes as no surprise that 9P is not a perfect match to the demands of other OS's IO systems. A simple example is Windows opportunistic locking. Under these situations, the client needs to perform an operation on a file where the operation is not supported by 9P. Usually at this point the client only knows the fid of the remote file.

Rather than regress to magic ioctl()s, we introduce the *extension* message – `Text` – and the respective response `Rext`.

`Text` is formatted the same as `Twrite`, `Rext` as `Rread`.

`Text`/`Rext` have been implemented in an Inferno[2] variant[4] as a testing platform. The kernel modifications were minor and a system call was added to perform a file extension request. To test the functionality *ramfs* was modified to respond to the `df` request with useful information and to reject others.

```
$ ramfs /tmp
...
$ fileop /tmp df
866304/1048576 82%
$ fileop . df
fileop: .: extension not supported
$ fileop /tmp colour
fileop: /tmp: unknown request
```

In order to eliminate a Tower of Babel problem with extension messages developed by different people, we define a common format for `Text` payload. We make the first byte the ID; we reserve two IDs, "**?**" – for simple query – and "**=**" – for ID query. The above example would have payload "?df".

[4]OzInferno.

The ID query is in preference to a master list of assigned IDs. In the following example, the agent asks the fileserver if it supports locking, and encryption.

```
Text  fid "=ntfs/oplock"
Rext "@"
Text fid "@lock params"
Rext "1234"
Text fid "=encryption"
Rerror "service not supported"
```

Once the ID is known then the payload, after the ID byte, is *vendor specific* – in the example case a formatted *oplock* request.

7.2. Scaling

Our estimates show that the maximum number of users on a typical Pentium-class server will be a function of its Internet bandwidth rather than its CPU or memory usage. As the users of the service increase, the service will require load distribution. We plan to solve this by adding a post-authentication, pre-9P negotiation that could redirect the client to a different server.

We may need to modify some Plan 9 file servers to cope with bigger user count – like eliminating linear searches – but we do not anticipate this before the number of users reaches hundreds of thousands.

8. Summary

Plan 9's name space facilities, name space multiplexing and the uniform application of 9P have provided an excellent foundation for implementing Shared Name Spaces, a system for creating secure virtual networks. Plan 9 enables an elegant and efficient solution. A similar system implemented on Sun J2EE or Microsoft .NET would be larger, more complex, more costly and more likely to have defects.

As an operations platform, Plan 9 has proven to be extremely reliable. Kernel support for off-the-shelf x86 servers makes the system cost effective.

Given our limited time and resources, developing such a system in any other environment would not have been possible.

References

[1] Russ Cox. Executive Summary - Muxfs Performance. 9Netics Internal Memo, April 2005.

[2] Sean Dorward, Rob Pike, David Presotto, Dennis Ritchie, Howard Trickey, and Phil Winterbottom. The Inferno Operating System. *Bell Labs Technical Journal*, Winter 1997.

[3] IBM Austin Research Lab Eric Van Hensbergen and Los Alamos National Labs Ron Minnich. Grave Robbers from Outer Space, Using 9P2000 Under Linux. In *Proceedings of the 2005 USENIX Annual Technical Conference*, Anaheim, CA, USA, April 2005. USENIX.

[4] Rob Pike, Dave Presotto, Ken Thompson, and Howard Trickey. Plan 9 from Bell Labs. In *Proceedings of the Summer 1990 UKUUG Conference*, pages 1–9, London, July 1990.

[5] D. Presotto and P. Winterbottom. The Organization of Networks in Plan 9. In *USENIX Association. Proceedings of the Winter 1993 USENIX Conference*, pages 271–280 (of x + 530), Berkeley, CA, USA, 1993. USENIX.

[6] Mark Weiser. The Computer for the Twenty-First Century. *Scientific American*, pages 94+, September 1991.