NetFS – Network Configuration Through the File System

Panagiotis Galiotos
USC/ISI
4676 Admiralty Way
Marina Del Ray, CA 90292
galiotos@isi.edu

Joseph Touch
USC/ISI
4676 Admiralty Way
Marina Del Ray, CA 90292
touch@isi.edu

Joshua Train
USC
Department of Electrical Engineering
Los Angeles, CA 90089
train@usc.edu

ABSTRACT

NetFS is a unified, platform-independent interface for network configuration using the file system and its operations. It introduces a novel approach where the system's network configuration components are mapped to a virtual file system. After mounting NetFS, network configuration can be performed by using familiar file system manipulations. Network configuration is unified and simplified while current configuration methods are still supported. NetFS exploits existing file system to offer fine-grained access control in which different permissions and policies can be granted to different users or groups. NetFS also uses remotely mounts to enable remote network management. By providing user-based virtual views of the system's network, NetFS can support network virtualization. Finally, integrating NetFS in configuration systems such as the X-Bone simplifies the configuration and control of overlays and VPNs.

1. INTRODUCTION

Network configuration is currently performed using a variety of different and non-portable APIs, thus requiring the user to become familiar with platform specific variants. For example, to setup a tunnel the FreeBSD user executes *ifconfig gif*, while the Linux user executes, *ip tunnel add*.

Further, existing command interfaces can be inadequate for advanced networking applications such as overlays or VPNs. Virtual networks and their applications need more sophisticated configuration tools in order to provide higher degree of control and security. For example, one application that runs over an overlay may need to reconfigure some network settings without affecting other applications or compromising the network's security. However, providing different applications or users with *root* privileges in order to perform this configuration exposes the entire system's configuration to that user, opening the whole system to vulnerability. Similarly, different users of a system may want to have different views of the network and be able to make changes that are transparent to other users. Again, visibility can be a security issue and permission restrictions minimize their flexibility. Finally, new features such as remote network configuration would ideally be part of a unified, extensible interface instead of requiring new tools and demand long learning curves of the users.

This material is based upon work supported by the National Science Foundation under Grant No. <NetFS:ANI-0129689>. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

2. NETFS MAIN CHARACTERISTICS

NetFS uses the file system interface as an API to the existing network configuration. It is a virtual file system that maps the system's network components to directories and files. Network interfaces and routing entries become directories and their attributes become subdirectories or files. Users simply execute the familiar file operations (e.g., *cat* into a file, or *mkdir*) to configure network parameters [1]. For example, setting the IP address of the Ethernet interface to 192.168.0.2 is done by executing:

mkdir 192.168.0.2

The advantage of this approach is a unified API that simplifies network configuration, thus minimizing learning time. Moreover, backwards compatibility is guaranteed as NetFS operates in parallel with the already existing interfaces without overriding them.

The NetFS architecture is based on implementing a subset of the VNODE operations [4]. Also by appropriately modifying some of them we can extend the NetFS functionality and provide the user with new features. Thus, NetFS reuses file system ownership and permissions mechanisms (user level

permissions) to control and distribute access capabilities to different users and groups for the system's network interfaces and routes. In that way each user can perform its own network configuration without affecting others or asking for *root* access. This enables NetFS to operate as a permission partitioning framework.

Network virtualization can also be simplified when using NetFS. Each user can have a customized view of the network components on a system instead of accessing the whole structure with the *root* access. Thus when executing *ls /netfs/ifaces*, one user may be able to list one set of network interfaces (e.g., *em0 lo0*), while another user lists different ones (e.g., *em0*). At the same time the system administrator can view all the devices (e.g., in this case, *em0 em1 lo0*).

Different routing entries can also be given to users trying to check the system's default gateway. For example, by executing *cat netfs/routes/gateway* one user may see *192.168.1.1* while another user sees *10.0.0.4*

NetFS also takes advantage of the file system's structure. The familiar directory and file hierarchy thus becomes a network component hierarchy. Based on the importance or the frequency of use, different components and their parameters can be placed on different hierarchy levels. This structure can significantly simplify the user's view of the network. For example, the directory called *ifaces* contains all the interfaces of the system. The subdirectory with the name *IP4address* contains the IP address of each particular interface. The user simply needs to navigate through this structure in order to perform the desired configurations.

NetFS is a virtual file system and hence it can be remotely mounted via NFS. NetFS thus becomes a remote network configuration tool. Of course, the installation must prevent certain pathological operations that would break the network path of the mount during such operations (such as changing the IP address over which the mount occurs).

3. CONCLUSIONS AND FUTURE WORK

NetFS offers advantages such as permission partitioning and the differentiation of network views between different users and groups. Overlay management systems, such as the X-Bone, can incorporate NetFS as a partitioning framework to provide limited networking control to different applications that run on different overlays. NetFS's unified interface can increase portability by eliminating the need for many different APIs for network control.

A future goal is to develop NetFS using the remote configuration and the permission partitioning features, notably to integrate it with the X-Bone [2] and other network configuration systems, such as IMUNES [3].

We intend to further develop the NetFS interface for other UNIX systems as well. Currently we are working towards the integration of additional network components in our implementation, including sockets. Lastly we note that NetFS needs constant updates and maintenance as it is an interface developed for the OS networking configuration subsystem which undergoes continuous changes.

Overall, NetFS provides a network configuration interface that integrates and simplifies already existing methods while it offers many additional features by exploiting the existing characteristics of the file system in favor of the network configuration. Hence, it provides permission partitioning and fine-grained access control. Network virtualization is supported either per process or per user. Finally by reusing the file system's remote mounting ability, NetFS also supports remote network configuration.

NetFS's implementation as a loadable kernel module (FreeBSD KLM) makes it a flexible solution that is easy to upgrade and modify without affecting the whole operating system. Finally NetFS provides backwards compatibility for a smooth transition to the new network configuration API.

NetFS is implemented as a Kernel Loadable Module (KLM) and is available for FreeBSD 5.4. For latest information and further details please visit our website at http://www.isi.edu/netfs.

4. REFERENCES

[1] P. Galiotos, J. Touch, J. Train, NetFS man pages.

[2] J. Touch, Y. Wang, V. Pingali, L. Eggart, R. Zhou, G. Finn., "A Global X-Bone for Network Experiments", Proc. IEEE Tridentcom 2005, Trento Italy, Mar. 2005, pp.194-203.

[3] M. Zec, M. Mikuc, "Operating System Support for Integrated Network Emulation in IMUNES", ASPLOS-XI, Boston, October 2004.

[4] R. Sandberg, D. Goldberg, S. Kleiman, Dan Walsh, B. Lyon, "Design and Implementation of the Sun Network File System", USENIX, June 1985.