Snowflake Work-In-Progress Report

Jon Howell
David Kotz

Department of Computer Science
Dartmouth College
Hanover, NH 03755-3510
{jonh,dfk}@cs.dartmouth.edu

August 19, 1997

The goal of the Snowflake system project is to learn how to best facilitate the construction of component-based distributed software systems that allow users to use resources across administrative domains without administrator intervention. Our current research is focused on namespace interfaces and the conventions used by namespace clients to locate resources. The three characteristics that set apart Snowflake namespaces are unification of disparate namespaces, object-oriented interfaces to named objects, and the absence of distinguished names such as the Unix filesystem root.

The first characteristic of the Snowflake namespace is the design of a general namespace that incorporates the many different namespaces used in traditional systems into a common space. Examples of objects traditionally accessed using names in disparate namespaces include files in the filesystem, hostnames in DNS, network interfaces in a list accessed by ifconfig, printers in `/etc/printcap`, processes in the process table, and configuration parameters in environment variables.

Secondly, Snowflake objects can communicate with other objects located via the unified namespace using object-oriented messages. The Plan 9 namespace went a long way toward a unified namespace, but objects located via the unified namespace communicate via a traditional filesystem read/write interface, which is awkward for many object types.

The third characteristic is that Snowflake avoids declaring any distinguished namespaces on behalf of component objects. In traditional systems, programs frequently look for resources using absolute paths, and those root-based paths can only be changed by the system administrator. Snowflake components conventionally look for any named resources starting at a user-supplied namespace root, so the user has ultimate control of the resources used by any component. This gives users freedom to transparently access resources from machines outside the administrative domain, or to confine components to resource "sandboxes." Also, traditional software often provides several ad-hoc and redundant ways to set configuration information, such as an absolute path to a system-wide configuration, a file in the home directory for per-user configuration, and environment variables for per-invocation configuration. In Snowflake, components need only look up a single name for configuration information, because the user is always capable of changing any name binding presented to the component, on a per-user or even per-invocation basis.

Our current work includes a Java implementation of a namespace with the above characteristics, a gateway to allow Unix programs to use the Java-based namespace as though it were a filesystem, and a gateway to map Unix files and other resources into the Java namespace. We will run some existing applications using these emulation gateways to understand how to exploit the functionality of the Snowflake namespace in new applications.

This work is supported by a grant from the USENIX Association and by a donation from Sun Microsystems.