SX.canary: A Network Diagnostic Tool for PCs

Semester Project
CS 522

John E. Harvey, Jr.
12/9/2000

Introduction

NxTrend Technology Inc. develops software solutions for warehouses and distributors. The software relies on remote access to a central database. Whenever two computers need to communicate, some sort of network must be used. The network must meet certain minimum requirements in terms of speed, reliability, and services in order for the software to function properly.

Networks come in many varieties. When one considers the number of products available and the possible combinations of those products, one would soon realize that it would be impossible to list every possible network situation. Also, every organizations needs are different when one considers their number of users and the location of those users. Therefore, NxTrend did not want to specify to their customers specific networking configuration requirements. Instead, the customer is allowed to use whatever network they feel they need as long as it meets minimum specified requirements.

However, this flexibility causes certain difficulties. First, for already-existing networks, how does the customer know if their network is able to support the product? Second, how are network related issues identified without detailed experience with a particular network?

The answer is called SX.canary. Canary is an application that runs on a Windows-based PC that tests the network that PC is connected to. The goal is to have it test all of the minimum requirements needed and report back to the user the results of the tests. Other requirements include a user-interface that allows the customer to run the program without

any technical knowledge. Also, the program should record test results so that they can be sent to NxTrend for analysis if needed.

At this time, SX.canary works in a basic sense. The user-interface framework is in place as well as a logging mechanism. It performs five tests: network socket support, host name resolution, reverse DNS resolution, ping connectivity, and effective bandwidth. Other tests will be added in the future, but the basic program will function as is. Therefore, it does work, but it is not complete.

Background Information

The Product Environment

Nxtrend's flagship product, SX.enterprise, utilizes an n-tiered architecture. At the center is a Unix-based computer that houses the database. The next tier is a Windows NT Server system. This system is called a Staging Client. It is a master client that is used to manage the software on other clients. The next tier is also a Windows NT Server and is called a code server. A code server manages the clients at a remote office. The last tier is called a Network Client. It is also a Windows-based system, but it does not have to be Windows NT Server. These are thin clients that don't have much software on them.

There are many relationships between the various systems. All of the clients need to access the database on the Unix system. All of the Network Clients must also be able to access either the Staging Client or a Code Server. Access to the database is accomplished using socket connections. Access between clients is accomplished using Windows Networking and shared disks.

There are many functions and services that require communication between the systems. The first and most basic is a simple (single record) query to the database. In this case, the client makes a connection directly to a database server process using a socket connection. The next case is a more complex query against the database that involves many records.

These queries are actually handled by a different process called an Application Server. The Application Server is running on the same system as the database servers. The query is performed on one machine without large amounts of intermediate results being transferred over the network. In this case, the client makes a socket connection to the Application Server to transfer data. Communication also happens between a Network Client and the Staging Client or a Code Server to get programs to run. When a Network Client needs to run a program that is missing from its cache, it must get the program from the Staging Client or a Code Server. This is handled through a filesystem "share" using Windows Networking. The final scenario is when SX.enterprise is being installed or updated. This is the primary function of the Staging Client. The software is first installed there. During the installation, a staging area is setup and that area is shared. Other clients can then access the installation files through that share on the Staging Client. During the software update process, there is an additional step where patched programs are also transferred to the Unix system from the Staging Client using FTP.

Network Requirements

In order for SX.enterprise to operate properly, the appropriate communications must be able to take place between the various systems. Therefore, clients need to connect to the Unix host, and they need to connect to drive shares of other clients.

Most networking applications make use of the Berkeley Sockets API. A socket is two endpoints of a communication which are each identified by an IP address and a port number [STEVENS98]. Unix-based computers make extensive use of Berkeley Sockets, and it is now standard on most (if not all) Unix distributions. Windows systems have more recently included support for sockets. Sockets are implemented through an API known as WinSock. The WinSock API provides the interface to the lower layers of the Windows networking system [QUINN96]. The earlier versions of WinSock were related to the Berkeley sockets, but not the same. The latest version of WinSock, however, has gone a long way to make Windows Sockets work and look the same as Berkeley Sockets. So, for a Windows-based system to make use of any kind of socket, it must have the