the Availability Digest

VoIP PBX Succumbs to Overconfiguration

June 2007

A software product development firm specializing in high-availability solutions received a real-life lesson in the very principles which it teaches. A fundamental precept of high-availability systems is that the bigger they are, the easier they break. The firm's new Voice over Internet (VoIP) PBX was overconfigured by the PBX vendor. The result – an early failure that didn't need to happen.[1]

The VoIP PBX System

The firm chose the open source Asterisk IP PBX software to provide its corporate phone services. Asterisk supports Voice over IP (VoIP), in which telephone conversations are carried over the Internet. Asterisk runs on a variety of operating systems, including Linux, Mac OS, OpenBSD, FreeBSD, and Sun Solaris. It can interoperate with most standards-based telephony equipment using relatively inexpensive hardware such as industry standard blades.

Asterisk is supported by the Asterisk Community, a worldwide group of volunteer developers. It is released as open source under the GNU General Public License (GPL) and is available as a free download.

The firm used a telecommunications vendor to design and implement its PBX. A Dell server blade was chosen to be the phone server which runs Asterisk. The server is front-ended with a ShoreWall firewall.

Two T1 lines connect the PBX server with the outside world:

- A T1 line provided by Siemens provides IP access to the PBX over the Internet using VoIP. Remote corporate VoIP office phones use this connection to get into the system.

- A T1 line from Verizon connects the PBX to the local telephone system. This connection provides twenty-three voice channels to the outside world.

Local office VoIP phones connect to the server via Power over Ethernet (PoE) switches, which supply power to these phones. In this way, the phones are powered by a single local power supply.

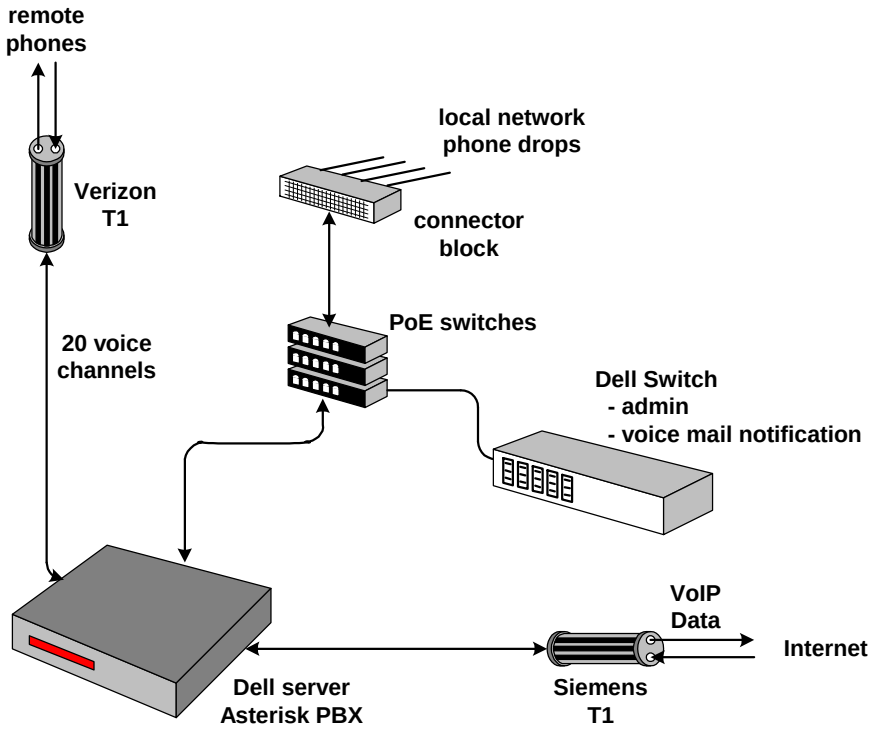
The system is currently supporting 40 office phones and ten remote phones. All corporate VoIP telephones, both local and remote, use the same corporate network.

All telephone traffic is routed through the Asterisk server. Incoming calls may come from local office VoIP phones via the PoE switches, from remote corporate VoIP phones via the Internet

[1] This Never Again story was contributed by Tristan Hoffmann.

connection provided by the Siemens T1 link, or from the telephone network via the Verizon PRI T1 link.

Each incoming call is routed by the server to the appropriate outgoing channel. Calls to the office phones are routed through the PoE switches. Calls to remote corporate phones are routed over the Siemens T1 Internet channel. Calls to external telephones are routed to Verizon via the Verizon PRI T1 channel.



The Asterisk Phone Server

The Asterisk PBX software runs on a Dell industry standard server in a blade configuration. The Dell server is configured with two 3-gigahertz dual-core Xeon 64-bit processors, one gigabyte of RAM per processor, a mirrored pair of 73 gigabyte disk drives, and a redundant power supply.

Since the server comprises two dual-core processors, it has available for processing a total of four CPUs. The various Asterisk processes are distributed among these four processors for load sharing.

The server runs the Linux operating system. It has no other third-party software except for QueueMetrics, a program to track information regarding call queues.

Asterisk can be configured through a Web Interface.

With multiple CPUs, mirrored disks, and redundant power supplies, the system is configured to be highly available. Right? Wrong!

The Crash

Just months after the initial installation of the PBX, the phones went dead. Nothing worked. Incoming calls were blocked. No outgoing calls could be made. The firm's phones could not even call each other over the internal IP network.

A quick investigation showed that the LCD display on the front of the server was scrolling an error message. Researching the error message, the firm's technical staff determined that one of the server CPUs was having an issue. A diagnostic monitor was connected to the server to get more detailed information. This pointed to a failure of CPU 4.

The staff first tried powering off the server and rebooting it, but the processor failure persisted. They then called the vendor technical support line and were authorized to remove the second dual-core processor from the server's motherboard (the one with the bad cpu in it). This time, after the system was rebooted, the server performed properly and continued to do so until after the weekend when a replacement processor was received and installed.

Post-Crash Analysis

Upon analyzing the crash, it became clear that the Asterisk PBX software and system BIOS made full use of all processors in the server. Its various processes were distributed among these processors to achieve load sharing and maximum capacity. Unfortunately, Asterisk was incapable of redistributing its processing load to the set of surviving processors in the event of a processor failure. Therefore, the processes assigned to the failed processor could not run and caused the crash of the server.

Interestingly, the firm found that the PBX ran fine on just one dual-core processor – proving that a "split system" configuration would be ideal if only the software and hardware supported that type of configuration. The server had been greatly overconfigured for the load which it was required to handle. By overconfiguring the server with two dual-core processors, both of which were used for system operation when only one was needed, the vendor had not-so-cleverly increased the probability of system failure by a factor of two (because it doubled the number of failure modes in the system). The system was only half as reliable as it could have been.

An obvious correction is to run with only one dual-core processor in the server. The server capacity is more than sufficient, and the server reliability will be doubled. Unfortunately, this violates the terms of the warranty; and the firm is precluded from doing this. The warranted use of this configuration is currently being negotiated.

Lessons Learned

There are important lessons to be learned from this experience.

1. Don't overconfigure your system with optional components that are in critical roles for operation but which do not add any meaningful function. In this case, two dual-core processors were used when one would have been just fine. Since the system made use of all processors available and was unable to configure around a failed processor, there were now two processor failure modes. The system would fail if either of the dual-core processors failed, thus doubling the system failure rate over that which would have been achieved if only a single dual-core processor had been used.

2. The firm was fortunate in that it is a software development firm and had on-site the skills to perform first-line maintenance. As a result, the system was down for just an hour and a half. If these skills were not available, the firm would have had to wait until a service

technician could be dispatched by the vendor. This would have taken an estimated eight hours or more before the PBX was once again functional.

System availability is the proportion of time that the system is up:

$$\text{Availability} = \frac{\text{uptime}}{\text{total time}} = \frac{\text{MTBF} - \text{MTR}}{\text{MTBF}} = 1 - \frac{\text{MTR}}{\text{MTBF}}$$

where MTBF is the mean time between failures and MTR is the mean time to repair, or the average downtime of the system following a failure. The term MTR/MTBF is the probability of system failure; it is the proportion of time that the system is down.

In this case, the system MTBF was half of what it could have been because of the use of unnecessary components in critical roles. However, the system MTR was greatly reduced by the on-site presence of skilled personnel.

The best use of the second dual-core processor would be to let it sit on the shelf as a spare to be immediately available should the operational processor fail. This would not only reduce the probability of system failure, but it would also significantly reduce the time required to correct the failure since there would be no delay waiting for a replacement processor to be delivered.