International Partners with Multi-Site Thin Client Interconnectivity

Brendan Conboy, Raytheon

Abstract. Do you remember when maintaining a code base with a foreign customer with development and test in both countries meant costly travel, restless nights due to hotel beds and time changes? Well we found a partial solution to this. Imagine being able to work on not only the code base, but virtually in the lab of your foreign customer without having to get on a plane, bus or ship, and enjoying the comfort of your own bed at night. With thin client and VPN technology we were able to save on time and travel costs and be able to more tightly integrate software and system changes. Now we still have language and culture boundaries, but those are out of scope with this article. With the ever evolving virtualization technologies the possibilities are multiplying daily.

Our story starts with an engineer that sees an opportunity to help his colleagues with work/life balance by allowing them to work from home easier with a thin client while tending to a sick child or watching a plumber fix their sink. This led to the idea, well if it works over the internet locally, why shouldn't it work internationally? Well after a long struggle with IT, firewall rules were created and properly tuned to allow the thin clients to exist inside our walls but not necessarily networked internally to maintain security. Now instead of planning a trip, booking tickets, travelling, adjusting to a new time zone, we can sit at our own desks, make a couple phone calls and connect in to another network. This even allows access to the development and integration labs allowing more efficient debugging with actual live site data.

Can software development be considered tactical? Does the development have an objective, or at least those developing it? Is there a schedule the software development needs to meet? Can a jointly developed software code base between two companies happen in "real time?" Perhaps not completely, however when the code base affects air traffic control and budgets and schedules are tight, it may be closer to those (real time and tactical) than not. When this program started in the very early 1990's the concept of a shared code base that could be developed by both parties in near real time was not even a thought. The best solution at the time was for customer engineers to travel to the contractor's site for initial requirements development and software architecture. Then over time software source code and executables were delivered on tape by hand. Once the system was accepted by the customer and the software warrantee period expired, then all software development moved to the customer's site. This is as far from real time tactical as we get in this article.

About a decade later this software development relationship took a big step forward. With both sides using what is now IBM's ClearCase software configuration management software, a multi-site relationship was established for a follow on contract and system upgrade. Code updates (what IBM calls synchronization [synch] packets) were encrypted using Gnu Privacy Guard (GPG) software and then exchanged using a now retired File Transfer Protocol (FTP) dropbox. These packets were eventually exchanged every week day via cron tasks on both sides. (Timing was an interesting adventure due to differing time zones and daylight saving time schedules) Now that the code base is synchronized in near real time, what about human communication. Well that is a little harder without a common language; at least the code was all in C. Even though both sides spoke English communication had troubles at times and still a lot of travel was required to either truly test or truly understand new features or integration issues, especially those "you have to see this to believe it…" kind of problems.

Now to introduce the thin client part of the story. The customer site had migrated to a thin client infrastructure for software development. This meant thin clients on developer's desks as well as in the lab for integration and debugging. By default, the thin clients used broadcast traffic to find a server and provide a "window" onto the network. What worked incredibly well as the "window" the developer had at their desk was the same view in the lab. Now take that a little further and consider the situation where the developer is at home either due an appointment or a sick spouse or child. Wouldn't it be great to have that same "window" from home? This would help the developer save valuable vacation time and maintain schedule for the company. So through the use of company supplied SecurID tokens, a corporate VPN concentrator and a version of the thin client's firmware this solution was realized. This network plumbing provides the necessary infrastructure for our ultimate use of thin clients for near real-time development and communication between the two companies.

Both companies were used to the travel and associated paperwork and business went on as usual. Now not everyone has the same schedule nor is available to travel any time or for any length of time. Also in addition to software development the customer started leveraging off the contractor's knowledge of Unix, networking, and development environment infrastructure. This knowledge is best leveraged if one was located at the customer site. Since that wasn't a possibility another solution had to arise, right? This led to the idea "well if the thin clients can connect over the internet domestically, why shouldn't it be able to connect internationally?" One huge difference was that customer's employees already had a SecurID token to use. How do we issue tokens to foreign contractors? Also is it worth opening this development network up to the world; the development network that affects Air Traffic Control?

> Now instead of planning a trip, booking tickets, travelling, adjusting to a new time zone, the contractor engineers can sit at their own desks, make a couple phone calls and connect in to the customer's network.

Now the customer had some work to do. They had to figure out how to allow this access and still maintain its own security. So extra interfaces were added off the thin client servers and connected to a special VLAN. This allows not only special rules to be applied to these thin clients but also an easy way to segregate or disconnect just those thin clients coming from outside the customer's site. This VLAN runs through the customer's intranet bound for a VPN concentrator off their firewall. Then a single VPN account for the contractor was created that is shared and linked to a single RSA token that is carried by the "on call" technicians at the customer site. The process for the contractor to connect was to power on the thin client, enter the shared username and PIN, then call the "on call" technician to get the token output. This provides a simple way to connect and preserves security for the customer, as they are the final gate to access. This was later expanded to a secondary shared account, PIN and token that is held by the customer's network support center that operates 24/7 to better accommodate the time differences between the customer and contractor.

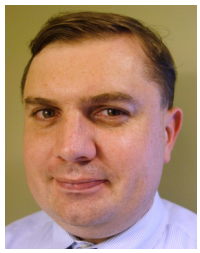
At the contractor site an isolated DMZ VLAN off the corporate firewall was created to house these thin clients. Since there is no permanent storage (besides some flash memory for the basic configuration of the device itself) in the thin client, security was a lot easier to maintain. Since nowadays VLANS can be extended to where ever one needs them. Thin clients were strategically deployed to the contractor's lab, users' desks and even common areas to accommodate turnover and even customer visits. Even customer visitors take for granted having their own desktop when they were at the contractor's site.

Now instead of planning a trip, booking tickets, travelling, adjusting to a new time zone, the contractor engineers can sit at their own desks, make a couple phone calls and connect in to the customer's network. This even allows access to the development and integration labs allowing for more efficient debugging with actual live site data. This was made available with the introduction and advancements in virtual frame buffer technologies (VNC and variants). Now while still at the contractor site one can log into the various test strings (even book the test strings like a customer employee) load them with your software and see the output, even interact with them, with live site data not available at the contractor's site. This is an incredible advancement since the customer site has test strings that emulate the depth and breadth of the fielded systems and includes actual live data feeds and other data inputs. This is a stark contrast to the contractor's site which has a bare bones system with a minimal subset of the equipment and only test data generators that simulate the live site data.

So from manual hand carrying media on planes and sleeping in hotels to staying home and getting to see the kids' sports games. We have come a long way in improving not only the speed and agility of our shared software development program. Not only saving schedule and travel expenses, but also helping balance out employees' lives. Will this work in every situation? You may be surprised. I was a part of it and cannot believe what we were able to do with as little effort as we needed.

Disclaimer:

ABOUT THE AUTHOR



Brendan Conboy is a Unix systems administrator at Raytheon Company in Marlborough, MA. He supports many programs and wears even more hats. Brendan joined Raytheon 1997 most of that time in Marlborough, but a couple years in St. Petersburg, FL. He is mostly supporting Air Traffic Control programs specializing in OS automation.

Special thanks to Einar Skagen of Avinor, AS in Norway who was the real architect and drive behind this change.

E-mail:
Brendan_E_Conboy@raytheon.com