ON THE SHIPBOARD APPLICATION OF NETWORK TIME PROTOCOL (NTP, RFC 1305)

Melanie E. Bautista
Lockheed Martin Advanced Technology Labs
1 Federal Street, M.S. A&E3W, Camden, New Jersey 08102, USA
Phone: (609)338-3987, fax: (609)338-4144, email: mbautist@atl.lmco.com

Abstract

The rapid evolution of naval combat system computing platforms is inspiring novel approaches in the design of shipboard time distribution systems. The Network Time Protocol (NTP) is being considered as a component in some of these new models. This paper presents studies conducted to evaluate the viability of NTP, in general, as a shipboard timekeeping mechanism, and in particular, as a non-mission critical timekeeping mechanism for the AEGIS Weapon System.

1 INTRODUCTION

The AEGIS Weapon System (AWS) is a complex system of systems. It is deployed on U.S. Navy cruisers and destroyers, and is capable of engaging enemy ships, planes, and submarines simultaneously. Defense Secretary William Perry has described the AEGIS Weapon System as providing "...thorough and accurate battlefield awareness by drawing intelligence about the battlefield from many sources and speeding it to the commander fighting the battle."

Traditionally, the computing base of the AWS consisted of military standard computers. These computers were programmed to perform a single function and communicated with other computers through direct connections, implemented as military standard low-level serial links. The traditional architecture utilized shadow processors, with primary and alternate computers, to provide fault tolerance. A dedicated military standard time distribution network enabled coordination among the various computers.

Today, the combined effect of changing threat scenarios, modification of doctrine to address these threats, injection of commercial-off-the-shelf (COTS) technologies, and diminishing defense spending is changing naval system computing architectures.

A migration towards a COTS network of high-powered COTS processing nodes, which hosts a suite of fully distributed application programs, is currently taking shape. In this new type of environment, by its very definition, the need to achieve system wide time synchronization still exists. The U.S. Navy is considering a number of novel solutions, some of which include the Network Time Protocol (NTP) as a component.

This paper presents a series of experiments conducted to evaluate the performance of NTP in a shipboard environment, as part of the fifth Engineering Development Model (EDM5) AEGIS Weapon System (AWS) communications benchmarking effort.

2 EDM5 LABORATORY CONFIGURATION

A laboratory test-bed network was constructed to support AWS EDM5 communications testing. The network was designed to simulate key processing elements, interconnected via candidate local area network (LAN) technologies and capable of generating representative levels of AWS communications traffic. The network provided the infrastructure and hardware to conduct a variety of computer communication benchmarks, including those for NTP. It consisted of nine Hewlett Packard (HP) workstations, three of which were Tactical Advanced Computer 3 (TAC3) HP-750 machines running the HP-UX version 9.05 operating system. The remaining six machines were TAC4 HP-J210's running HP-UX 10.01. All nodes supported two interfaces: an Ethernet interface to a file server and a Fiber Distributed Data Interface (FDDI) to a router backbone. The router backbone consisted of four Cisco 7513 routers interconnected in a fault-tolerant FDDI configuration.

3 SHIPBOARD TIME SYNCHRONIZATION REQUIREMENTS

The requirements for time synchronization in a heterogeneous, diverse, networked collection of computers take added dimension when the target operating environment is aboard a battle-ready ship. Whereas a land-based commercial application network might concern itself primarily with the accuracy, portability, and scalability of a timekeeping mechanism, a sea-based tactical application network must also guarantee the settling time, stability, and survivability, as well as more stringent synchronization requirements, of its timekeeper.

Synchronization describes the ability of NTP to make a system of clocks agree in both frequency and time. Settling time describes the elapsed time required for NTP to reach nominal synchronization. Stability describes the ability of NTP to regulate the clocks, such that measured time offsets do not climb or descend too far or too often. Survivability describes the ability of NTP to be robust in face of system failures or degraded system states.

4 EXPERIMENTAL OBJECTIVE

Performance requirements in each of these categories are derived from the AWS A-level Specification, and differ for mission critical versus non-mission critical elements. The goal of EDM5 NTP testing was to evaluate NTP against these performance criteria in the non-mission critical configurations currently planned, and to gain insight into how NTP might perform in a mission critical configuration.

5 NTP EXPERIMENTS

Because NTP offers a dynamic clock correction mechanism, its behavior is expected to vary with time. Because it is significant to understand such behavior over both the short- and the long-term, two experiments were designed and conducted.

5.1 NTP Stability Experiment

5.1.1 Objective

The NTP Stability Experiment was designed to measure the long-run synchronization and stability of a system of clocks disciplined by NTP. The experiment was also designed to allow a measurement of initial settling time. The first part of this experiment configured a timekeeping subnetwork from the EDM5 test-bed equipment, establishing NTP communications over 10 megabit-per-second (Mbps) Ethernet. The second part of this experiment utilized the same configuration, but, instead, established communications over 155 Mbps FDDI. Figure 1 shows the NTP synchronization subnet topology exercised for both parts of the experiment.

5.1.2 Configuration

As shown, host wil13429 was designated as a stratum 10 NTP server, "master." Host wil13428 was designated as a backup NTP server, "backup master," at stratum 12. The master and backup master were then designated as peers. All other hosts in the test-bed were designated NTP clients to both masters. The master and backup master employ their own local clocks as time references. Therefore, this NTP configuration attempts to slave all clocks in the test-bed to the local clock of wil13429, and to wil13428 in case of failure.

5.1.3 Implementation

Automated statistics collection was enabled to capture NTP system state variables for the duration of each test; and the NTP synchronization subnet was permitted to free run relatively undisturbed. Part I examined clock behavior for 11 days. Part II studied behavior for 7.

5.2 NTP Load Experiment

5.2.1 Objective

The NTP Load Experiment was designed primarily to study the survivability aspects of NTP performance. It does this by subjecting NTP host computers to a harsh environment, and comparing clock synchronization and stability before and after stress conditions are applied. The experiment also tests for the impact of host computer crash failures and recoveries, which are simulated by abruptly terminating and re-initiating the NTP software on the NTP hosts under test.

The harsh environment consists of two components: contention for communication resources and processing resources. Communications resource drain was simulated by injecting a bursty traffic pattern into the network. Processing resource drain was accomplished by running CPU-intensive applications on the NTP hosts under test.

A series of test scenarios composes the experiment. These scenarios attempt to model conditions that might occur during shipboard operations. For example, it is conceivable for a situation to arise, in which NTP is able to run for only 15 minutes after start-up, before being terminated during a failed system state. It then remains inactive for 10 minutes before being re-instantiated. Yet, throughout its activity, NTP must compete for network and/or central processing unit (CPU) resources, due to the stress associated with system failure and recovery. How NTP will perform under these and similar conditions is of significant interest. Therefore, scenarios were carefully constructed to benchmark NTP as part of this experiment, as shown in Figure 3.

Because of the short duration of each test, the NTP Load Experiment also provided an opportunity to study short-run performance against synchronization, stability, and settling time

requirements.

5.2.2 Configuration

This experiment configured a basic timekeeping subnetwork consisting of a single client and single server from the EDM5 test-bed equipment. Communications were established over FDDI. Figure 2 illustrates the basic NTP synchronization subnet topology used. A TAC3 and a TAC4 versions of this configuration were implemented. In the TAC3 version, wil13428 acted as an NTP server to NTP client wil13427, with the local clock of wil13429 providing a time reference to both. In the TAC4 version, wil13430 acted as an NTP server to NTP client wil13435, with the local clock of wil13434 providing the time reference.

5.2.3 Implementation

The experiment was conducted as follows: First, NTP configuration files were modified to implement either the TAC3 or TAC4 synchronization subnet. Second, each scenario was executed in series as listed in Figure 3, constituting an iteration of a test suite. Third, the levels of network and processor activity were altered and the test suite rerun. Figure 4 outlines the NTP configurations and corresponding platform activity levels used for each iteration of a test suite. Completion of the experiment was signified by exhausting the list.

To implement the experiment, several UNIX scripts were developed. A high-level description of each is provided. The main script manages the initiation and termination of the NTP client and server as needed to match the sequence defined by the test suite. It invokes a separate script to calibrate the NTP client and server at the start of each test scenario. It also invokes the NTP query utility, "ntpdate," periodically to collect and record time offset statistics. The script was designed to execute a single iteration in 9 hours, at 1 hour per scenario, including 20 minutes for calibration.

The calibration script simply instantiates the NTP client and server together, lets them run for an amount of time, then terminates each just before the test scenario begins. The calibration phase attempts to equalize NTP performance in the client and the server hosts.

The network load scripts generate bursty traffic over the network by invoking an AWS communications simulation tool, developed in-house, in a predetermined sequence. In an attempt to affect the maximum network-related stress on NTP, this sequence specifies parameters which cause the tool to create different levels of network utilization approximately every minute for the duration of an iteration. Because the NTP algorithms choose an optimal source of timekeeping information partially as a function of past statistics on path delays, creating an environment in which path delays vary each time NTP communicates increases the likelihood that NTP makes a nonoptimal decision for the current network state. A FDDI Network Advisor was used to measure and record traffic traces during each loaded network iteration to verify the test conditions. Note that the network load scripts were executed manually on separate hosts from the NTP hosts under test.

Two versions of CPU load scripts were utilized for the experiment. One script, again, exercised the AWS communications simulation tool. This time the tool was invoked on each of the NTP hosts under test, such that the processors were constantly polling for a control message scheduled never to arrive. The other script simply repeatedly invoked a recursive UNIX "find" command. Both scripts drained approximately 96% of CPU resources, as measured by the UNIX utility, "top."

6 ASSUMPTIONS

A series of assumptions limit the scope of the experiments as designed:

The standard distribution of NTP is tested. This version of NTP is tuned for networks expecting to synchronize to clock references over the Internet. Note that the Naval Surface Warfare Center, Dahlgren Division (NSWC-DD), is studying which NTP parameters are tunable and how to tune them in order to optimize performance in an AEGIS-like network. Such a network is isolated from the Internet, is more geographically compact, and utilizes clock references local to the ship. An independent validation of the NSWC-DD findings is planned, as well as rerunning of the experiments described herein to characterize NTP when it has been appropriately tuned.

Only TAC3 and TAC4 NTP performance is characterized. Because the quantity and quality of synchronization achievable via NTP is ultimately determined by the hardware and operating system it attempts to discipline, the scope of any characterization of NTP is limited to the platform in question, and cannot necessarily be extrapolated to other host platforms. Specifically, mission critical elements are not being hosted in either the TAC3 or the TAC4 platforms. Therefore, it is not fully accurate to compare NTP performance, as measured by the experiments reported herein, against mission critical requirements.

Only the symmetric active mode of NTP operation is evaluated. This mode enables the maximum exchange of time information in the subnet and, therefore, theoretically produces the most reliable synchronization. Properties and performance of other NTP modes will not be studied in these tests.

Sufficient aging of the NTP drift files has occurred. The NTP algorithms are implemented as a software daemon, "xntpd." Because xntpd attempts to discipline time by regulating the oscillator aboard each processor, its performance enhances with increased understanding of the idiosyncrasies of the hardware. Thus, xntpd records drift data to a file, which is continually updated during runtime. A more veteran drift file provides better information to xntpd.

NTP accurately reports synchronization performance. The xntpd software distribution delivers two query utilities, "xntpdc" and ntpdate, for examining NTP system state variables. These variables include time offset between pairs of NTP nodes. NSWC-DD determined that xntpdc reports more dated information than ntpdate, but concluded that NTP does accurately measure its own performance. Based on these findings, it is most accurate to utilize ntpdate when conducting short-duration tests, whereas for long-duration tests, xntpdc still provides an accurate depiction of performance. This study makes an assumption that xntpd automated statistics collection employs techniques similar to xntpdc. This is a reasonable assumption, based on a cursory examination of the code.

7 STATUS

To date, all experiments described in this paper have been completed. The data have been collected, and much of them have been reduced, plotted, and analyzed. However, a full analysis, to produce measures of synchronization, settling time, stability, and survivability, is pending completion and will be published as a separate document.

8 PRELIMINARY CONCLUSIONS

As of the date of initial data analysis, AEGIS EDM5 system requirements were continuing to evolve towards final definition. Therefore, it was not possible to conclude with certainty whether NTP, when hosted on the TAC3 and TAC4 platforms, can satisfactorily perform the non-mission critical time synchronization function for an EDM5 incarnation of AEGIS. However, preliminary data suggest that, as expected, in the configurations tested, NTP cannot meet mission critical performance requirements. It is believed that adding a highly reliable time reference to these configurations might enhance performance enough to reverse this conclusion.

The suite of tests and evaluation did indicate that the movement in time and frequency of clocks, disciplined by NTP, occurs in a coordinated fashion. Furthermore, it was determined that when xntpd is terminated on a host, its local clock begins to drift immediately until the daemon is re-initiated. It is significant to note that NTP performance degraded when introduced onto the TAC4 platform. This confirms that operating system upgrades do not necessarily imply NTP performance upgrades. Instead, xntpd should be benchmarked for all candidate hardware platforms, operating system revisions, and synchronization subnet topologies.

Perhaps, the most important conclusion drawn from these experiments is that NTP performance can be significantly enhanced by tuning xntpd. By customizing user-definable parameters for an isolated, geographically localized synchronization subnet, it is expected that synchronization, settling time, stability, and survivability numbers will drop. It has been demonstrated that reducing the polling interval, from a 64 to 1024 second range to a 16 to 64 second range, trims client-to-server offsets by an order of magnitude. Continued testing using actual EDM5 hardware and introducing mission critical synchronization subnet topologies are recommended.

9 REFERENCES

M.E. Bautista, S. Dempsey, W.J. Reilly, M. Teter, and L. Weinberg, *"EDM5 network communications testing status report, February 1996,"* NS-C-ADV-T-2001, April 1996.

"High performance distributed computing program: engineering testbed one report," Naval Surface Warfare Center/Dahlgren Division, and Johns Hopkins University Applied Physics Laboratory, November 1995.

D.L. Mills, *"Network Time Protocol (version 3) specification, implementation and analysis,"* RFC 1305, March 1992.

K.F. O'Donoghue, and T.R. Plunkett 1996, *"Development and validation of network clock measurement techniques,"* Proceedings of the 4th International Workshop on Parallel and Distributed Real-Time Systems (IEEE Computer Society Press).

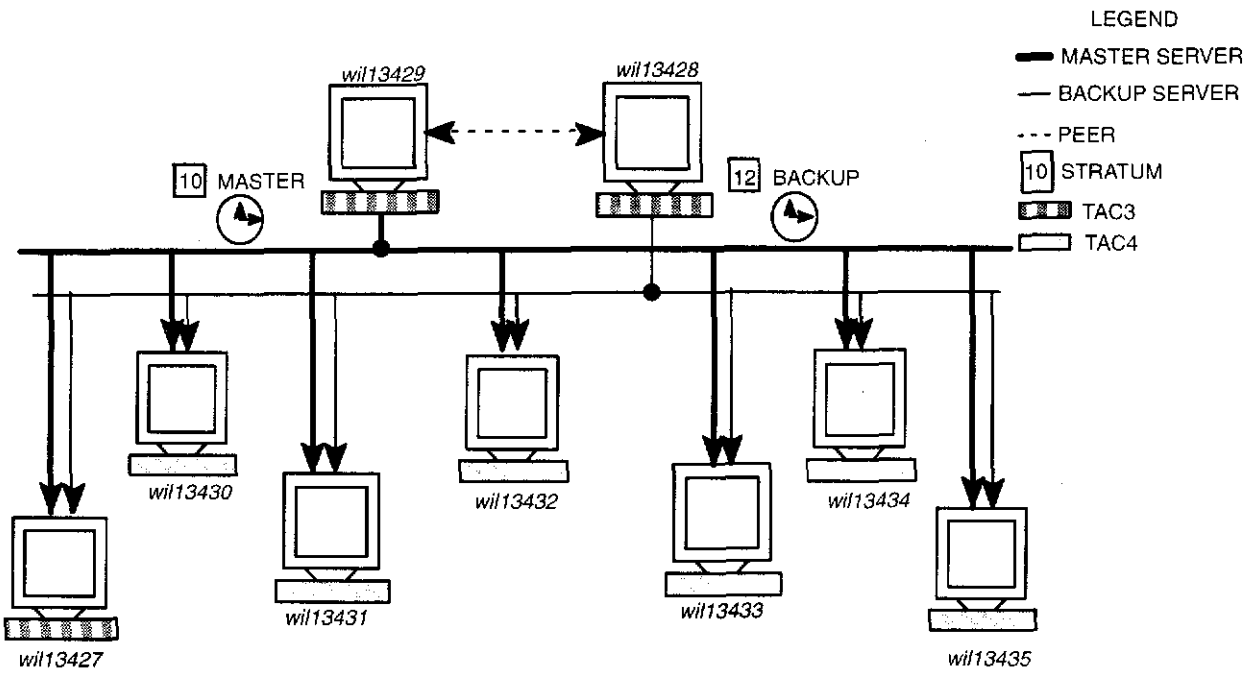

Figure 1. NTP Stability Experiment Synchronization Subnet Topology

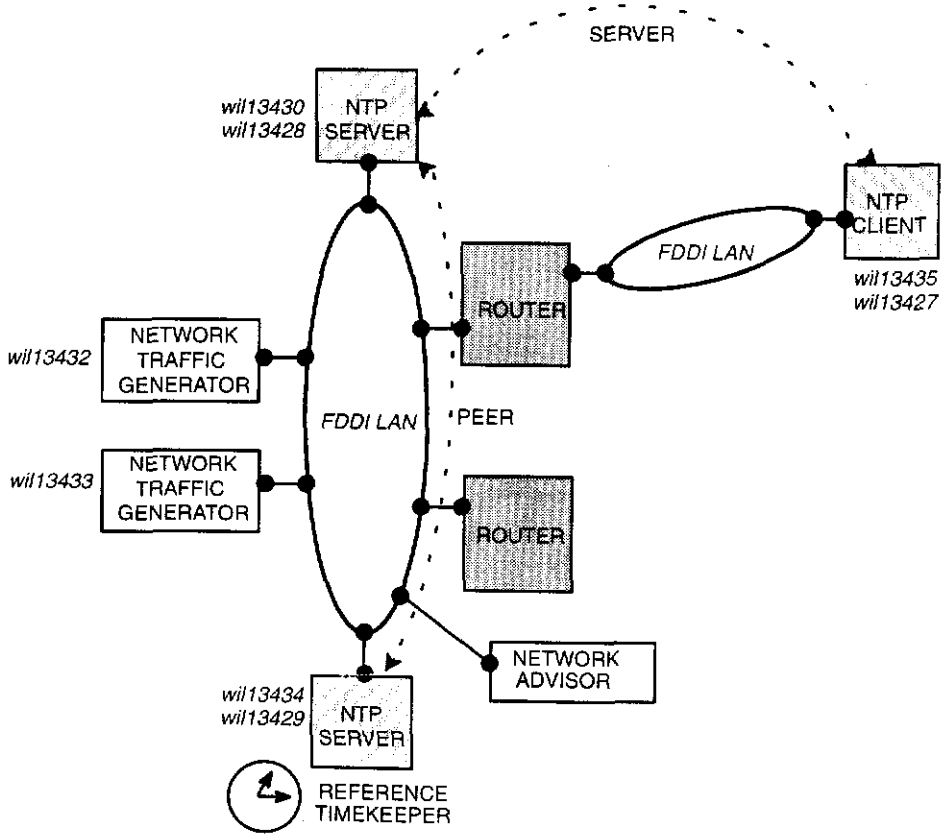

Figure 2. NTP Load Experiment Synchronization Subnet Configuration

NTP Client and Server Active Period Profiles Specified in Minutes	Reference Name
Client/Server Up Continuously	csuc
Client Up 10 Down 30	c1030
Client Up 20 Down 20	c2020
Client Up 15 Down 10 Up 15	c151015
Server Up 10 Down 30	s1030
Server Up 20 Down 20	s2020
Server Up 15 Down 10 Up 15	s151015
Client/Server Up 10 Down 15 Up 15	cs101515
Client/Server Up 15 Down 10 Up 15	cs151015

Figure 3. Scenarios Composing a Single Test Suite

	[64 1024] second NTP polling interval	[16 64] second NTP polling interval
TAC–3	no load	no load
TAC–3	network load	network load
TAC–3	CPU load	CPU load
TAC–3	network and CPU load	network and CPU load
TAC–4	no load	no load
TAC–4	network load	network load
TAC–4	CPU load	CPU load
TAC–4	network and CPU load	network and CPU load

Figure 4. NTP Configurations and Platform Conditions During Iteration of a Test Suite

Questions and Answers

RICHARD SCHMIDT (USNO): What version of NTP were you doing this test on?

MELANIE BAUTISTA: 3.4.

RICHARD SCHMIDT: And did you do any tests – it wasn't quite clear to me what was the effect of disabling the servers, letting the clients freewheel on NTP from their last known drift rates?

MELANIE BAUTISTA: Actually there were several iterations that were tried. The main iterations were actually disabling NTP on the client machines as well. So it was done in both directions. We disabled the daemon on just the client and then re-initiated and we observed how synchronization was affected. We also did the reverse.

RICHARD SCHMIDT: But I guess my question was we know if you kill NTP on the client, the client's going to run away. There's nothing controlling it. But if you disable access to the server, after some period of time it should have learned what its frequency errors were and it should freewheel fairly accurately. But that depends on how long NTP had been running before you kill that access. Have you done any tests showing that dependency, how long it needs to be running before the client can run fairly successfully without a server?

MELANIE BAUTISTA: Actually, we didn't extend the experiment to that extent. However, these experiments were designed to study short-run characteristics of NTP. So the tests were very short.

The first experiment examined NTP synchronization over a number of days. The NTP load experiment conducted nine trials in series, each trial lasting one hour. So it would be something like the client/server was on for 15 minutes together; client/NTP daemon was terminated for 10 minutes; then both came back on for 15 minutes. And then we recorded the time offsets and studied whether in that short time period we could learn anything worthwhile, because this is the type of environment we would be expecting in an AEGIS environment.

JUDAH LEVINE (NIST): I wouldn't want to talk anybody out of using NTP, but it seemed to me that most of the machinery that is built into NTP is really not needed in your environment. And most of the problems in your environment are things that NTP is not going to help with, in that you have a network and it is what it is. So that its delays are going to be whatever they're going to be. I mean, right, you have some topology and it is what it is. I guess the whole idea of estimating the network every time – I mean, you could just do it, right? It's your closed network. You own the whole thing. It's not like you have to go through anybody else's router.

Let me ask why did you choose NTP as opposed to all the other choices that were available?

MELANIE BAUTISTA: Well, the reason NTP was chosen is because it's automatic, it's something that you can turn on in your workstation, you never have to worry about it. It's true that you could do your own simple implementation of measuring the network delay and doing the simple calculation that NTP does without the additional overhead that NTP builds in in order to support an Internet environment. However, it's implemented; you can just turn the daemon on in your workstation and not worry about it.

The initial concept was not chosen for the mission-critical elements. It's more for all of the workstations on the ship performing non-mission-critical functions.

WILLIAM BOLLWERK (USNO): I have a question for you about the requirements. You say

that there were requirements, and you broke them down into four areas. Those came out of Operations Requirements Document for AEGIS, or where did they come from? The ORD – or what are the actual timing requirements and are they for the critical shipboard operations? You just mentioned that the system is set up for the nonessential, like the supply functions and other things on the ship. Are the requirements broken out into that type of category or what do you have there?

MELANIE BAUTISTA: Because this is really a new feature that's going to be introduced in future baselines of AEGIS, requirements for non-mission-critical time synchronization do not currently exist. However, the concepts being developed for future AEGIS have a different concept of operations, and many of these requirements are being derived from that. A lot of them, however, are based on the AWS top-level specifications such as first start-up times; you have a fixed amount of time that you have to be able to get the whole system up and running by "x" amount of time; if the system goes down, you have to come back alive, everything has to be running within "x" amount of time. Requirements on that level help to define the fault tolerance in fault recovery requirements for NTP.

In terms of synchronization, to date when we are testing, we are testing against old synchronization requirements, in other words, requirements made of the time distribution system in past AEGIS baseline designs. So what the previous system had to perform, we have to at least be able to meet those requirements and perhaps ...I'm not at liberty to say; that's classified information.

KAREN O'DONOGHUE (NAVAL SURFACE WARFARE CENTER): I just wanted to add a couple of things to what Melanie has already said. The question about why NTP might be useful onboard ship – first of all, I don't believe that we see the shipboard network being quite as static as might have been indicated. There are a number of routers onboard ship; there are a number of various subsystem networks; and I think we see the need to be able to – especially in a fault scenario where you're losing networks and various routers might be going down that we might need more flexibility then – we don't have a static network, I guess is what I'm trying to say.

One of the second motivations for looking at NTP is since we are moving towards using COT space workstations onboard ship, we are very interested in the capability of the frequency – being able to modify the phase and the frequency. Because, on COT space workstations you get the capability to figure out what the frequency offset is and to possibly make some correction for that is very attractive.

On the question of requirements, one of the things I'd like to refer to is that there were a number of efforts done by the Navy in the Next Generation Computer Resources Program, which has recently ended, looking at what the timing requirements were for networks, in particular, in the Safenet Program; and some of the earlier requirements documents that came out of that that were made public is that they wanted the synchronization between processors to be within 1 millisecond; and that you needed – I believe the term you used was "settling time," I'm not sure – to be within 5 seconds. So from the time the machine boots, within 5 seconds we need to be synchronized to within 1 millisecond and then maintain that level of stability. Thank you.