QoSYNC: Quality of Synchronization for Clocks on Networked Computers

Arthur de Castro Callado[1], Judith Kelner[1], Alejandro C. Frery[2]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE – Brazil

{acc2,jk}@cin.ufpe.br

[2]Departamento de Tecnologia da Informação – Universidade Federal de Alagoas
57072-970 – Maceió – AL – Brazil

frery@tci.ufal.br

Abstract. *The Network Time Protocol, an over 2-decade old and always improving algorithm for synchronizing networked computer clocks, still finds problems for its efficient operation. Many applications need a trustable time system to function correctly (e.g., banking and distributed database servers). With the advent of Quality of Service in computer networks, this problem can be elegantly approached and solved. This article suggests a framework for dealing with clock synchronization on DiffServ Domains, introduces a novel treatment to packets and validates this proposal on a case-study done with live application metrics in a network emulation environment.*

1. Introduction

Clock synchronization in computer networks is very important to ensure the proper functioning of systems, and is fundamental for the integrity of time-sensitive services.

The accuracy of clocks synchronized through computer networks depends on the algorithms used, on the delays the synchronization packets are submitted to and on the variation of such delays (the *jitter*). Therefore, guaranteeing minimum and stable delays when transporting these packets is a way of ensuring better clock precision. The most used clock synchronization algorithm, the Network Time Protocol (NTP), works over IP and has been accepted as the Internet "de facto" standard.

Some statistical techniques are used to yield an acceptable solution for the problem, but there had been no proposal to improve the quality of synchronization based on quality of service (QoS). With QoS it is possible to offer guarantees about delay and jitter to the synchronization mechanism, benefiting all the applications that depend on it.

In this work, we analyze of the benefits quality of service may bring to clock synchronization, including measuring QoS parameters on NTP. The 2nd section describes clock synchronization on Computer Networks with NTP. The 3rd section mentions some QoS architectures and details the Differentiated Services (Diffserv) architecture. The 4th section describes the QoSYNC Framework proposal for ensuring Quality of Synchronization for clocks over the Internet. The 5th section comments on the results and the 6th section brings conclusions and cites some future works.

2. Clock Synchronization in Computer Networks

The *Network Time Protocol* (NTP) [1][17], standardized by the Internet Engineering Task Force (IETF), has been the "de facto" standard for clock synchronization for over two decades and has been improved over time. Nowadays, a huge number of time-servers [17][23] around the world utilize it to replicate its time to other machines.

2.1. NTP Hierarchy

NTP uses a hierarchical and distributed algorithm to share time [16]. Physical clocks [15] are used as references of time and can be of any type. Atomic, radio wave-based and GPS-based (Global Positioning System) clocks are the most common. Each of these is attached to a computer called *stratum 1* [17]. These computers serve a very trustable time, but costly and require special skills to install and configure.

Any computer that synchronizes using a stratum 1 computer as a reference is called *stratum 2*, and may serve time to *stratum 3* ones and so on up to *stratum 15*, forming a synchronization tree. Therefore, the stratum measures the distance to a physical clock, in network nodes. Each computer has an internal clock (normally a quartz oscillator) to keep time, called local clock, which is less dependable but inexpensive, since the leaves the tree do not need a high time precision. User machines are commonly stratum 4. If an accurate machine is needed (e.g., for banking or distributed database servers) moving up in the synchronization tree is required as, for instance, connecting directly to a physical clock or synchronizing time to a stratum 1 through a low jitter path (one would better buy a good clock than pay the extra link). An unsynchronized computer is called *stratum 16*. The NTP interaction with a server is called *association* (it is not connection-oriented).

2.2. Clock Selection Algorithms and Synchronization Process

A computer is normally configured with many reference clocks, but utilizes statistical techniques along with polling to choose the best ones. A mathematical description of the problem can be found in [14] and some used clock-selection algorithms can be found in [14][16]. The stochastic process of clock synchronization is modeled in [14] and [5].

Synchronization requires periodic message exchanges between client and server. A single message exchange i is performed in 4 steps. First, the client generates a timestamp T_{1i} and includes it in a message to the server. Then, the server receives the message and generates another timestamp, T_{2i}. After user authentication and packet processing, the server generates another timestamp T_{3i} and sends a message back with all three timestamps. Finally, the client receives the responding message and generates another timestamp T_{4i}. The timestamps are used to compute estimations for clock offset δ and roundtrip time θ, as follows:

$$\delta_i = (T_{4i} - T_{1i}) - (T_{3i} - T_{2i}) \qquad \theta_i = \frac{(T_{2i} - T_{1i}) + (T_{3i} - T_{4i})}{2}$$

NTP feeds the clock discipline algorithm (compute corrections) with δ and θ. NTP can use authentication to prevent unauthorized access to a clock or generation of undesired network load [16]. This is beyond the scope of this paper.

2.3. Clock Discipline Algorithms

The clock discipline algorithm corrects the local clock (variable frequency oscillator - VFO), compensates for intrinsic frequency error and adjusts various parameters dynamically in response to measured network jitter and oscillator frequency stability (called oscillator *wander*). It is based on a feedback control system.

2.4. Quality of Synchronization

Some interesting (and often confusing) metrics can be used for time synchronization [16]. The *stability* of a clock is how well it can maintain a constant frequency, while *accuracy* is how well the frequency compares to time standards and *precision* is how precisely these quantities can be maintained on a system (maximum error estimation). The *offset* of two clocks (here, an NTP client and a server) is the time difference between them. The *skew* represents the frequency difference between them (computed as the first derivative of offset with time) and the *drift* is the variation of skew (second derivative of offset with time). Room temperature changes are the main causes of drift.

According to Mills [15] NTP yields better results (i.e., offset can be measured with the smallest error) when the network load is not high. That is not a problem because network links generally spend very little time with high loads, but during high and even low load periods, due to the lack of control over packets bursts that generate jitter (because of queuing) and loss, NTP loses synchrony and the machine depends solely on its local clock to keep the time. Then, it must wait until the network behavior stabilizes and NTP [15] synchronizes again (which takes many minutes).

Clock synchronization depends on the reference clock quality, local clock stability and network delay and jitter [16]. Therefore, by guaranteeing minimum and stable network delay for request packets one guarantees permanently good synchronization for computers that use NTP to synchronize their clocks. A simple application that easily shows the benefits of good clock synchronization is a network analyzer that measures one-way network metrics called *IPstat* [2].

3. Quality of Service

Many architectures for QoS provisioning have been proposed to deal with different network service requirements. The most important ones [26] are *Differentiated Services* (Diffserv) [3], *Integrated Services* (Intserv) [4] and *Multiprotocol Label Switching* (MPLS) [1]. Diffserv was chosen for the task of controlling such traffic since it is the most used and implemented architecture in academic networks, it is the easiest to implement in the Internet and it perfectly adapts to the service requirements.

3.1. The Differentiated Services Architecture

Diffserv was designed to have a smooth implementation in current Internet by being able to work with legacy applications and network topologies. Only router update is actually required and can be done on a per-network basis.

The IPv4 header has a byte-long field named "*Type of Service*" (TOS), designed to select different treatment to packets marked by applications, especially network control. The "IP Precedence" used 3 of the 8 bits so that applications could signal the

need for low delay, high throughput and low loss. The other bits were unused. The main idea in Diffserv is to map each configuration (a *Diffserv Code Point* - DSCP) of the TOS byte (now called "*DS Field*") to a different packet forwarding treatment, named *Per Hop Behavior* (PHB). Since the number of PHBs is limited by the size of the field, Diffserv routers treat only aggregated traffic flows, formed by a group of *microflows*.

3.1.1. Per Hop Behaviors

Some PHBs have been standardized, as explained below.

The *Best Effort* (BE) PHB is the current treatment used on the Internet. It delivers fairness (among packets of the same class) without guarantees. It was not specifically standardized for Diffserv, and its old definition from [21] is still valid.

The *Assured Forwarding* (AF) [7] is a *Behavior Aggregate* (BA) that offers discard priority guarantees within a class. There are 4 AF classes, each with 3 different drop precedence values, making a total of 12 DSCPs for AF. It can be used by an application with different traffic flows (with flows more important than others, e.g., layered video transmission) or by many applications with differentiated importance.

Expedited Forwarding (EF) [9] guarantees no loss and delays close to the minimum possible. EF traffic should be independent of other traffic on the same router.

3.1.2. Diffserv Network Topology

The Diffserv network topology may have four elements [26]. The most important is the *Core Router* (CR), which is a Diffserv-aware router that can treat packets with PHBs according to its DSCP markings. The *Edge Router* (ER) is a core router also responsible for packet classification, marking, policy enforcement and traffic shaping. It is generally used at the borders of a DS Domain. With so many tasks to perform, ERs are the bottleneck [26] of a Diffserv domain and must be well equipped (being, therefore, expensive). Packet markings can be trusted to applications, but since any network can use private markings (besides standard PHBs), some remarking may be necessary between DS Domains.

The *Bandwidth Broker* (BB) makes *Service Level Agreements* (SLAs) with network entities automatically to request services on-demand according to specified requirements (bandwidth, maximum delay/jitter, maximum drop probability, etc). Based on the obtained agreement it informs the affected routers within its domain of the necessary changes. A *DS Domain* is any network that can treat Diffserv traffic. It can also interact with other non-DS Domains, but then no guarantees can be made.

3.1.3. Service Level Agreements

The Diffserv architecture does not define a protocol for the applications to reserve any resources in Diffserv routers. The customer must have a *Service Level Agreement* (SLA) with its provider to specify the services required, the guarantees offered, the method of billing for the service and the contingencies and penalties for not respecting the guarantees. SLAs are also made between providers (*DS domains*). SLAs can be static (described by a signed document) or dynamic, through the use of BBs. Due to the aggregated treatment of traffic, Diffserv routers cannot make guarantees on microflows,

e.g., a drop guarantee of less than 3% for an AF service does not mean that every application that makes use of this service will experience less than 3% packet drops. It just guarantees that the average will be respected.

BBs are a tendency as well as a requirement for the acceptable use of a Diffserv network. When analyzing the necessary time for network provisioning, one realizes that waiting for people to agreeing and signing a document (even a digital document) is not a reliable way of doing provisioning or capacity planning. The cost of installing a better infrastructure than what is momentarily needed is surpassed by the costs of updating it.

4. QoSYNC Framework Proposal

In this paper we propose a configuration scheme for improving quality of clock synchronization (QoSYNC) with the use of NTP and the Diffserv architecture.

To our knowledge there has been no work on the analysis of the performance of synchronization algorithms on the light of quality of service architectures.

4.1. Diffserv Configuration

In a network that will serve NTP traffic, this work compares the use of two PHBs: the Expedited Forwarding or our proposed Hot-Potato Forwarding described next.

4.1.1. The Expedited Forwarding PHB

The EF PHB can be used to deliver NTP packets. Due to its efforts for providing minimum delay and no loss NTP would have good stability. But NTP does not need the loss guarantees of EF and the delay guarantees could be slightly improved. In spite of the guaranteed bandwidth of outgoing EF traffic being big enough to accommodate all the incoming EF traffic [9], due to simultaneous arrivals and small bursts it is common to have a little queuing of EF traffic. This can hurt synchronization.

In order to configure the routers to transport NTP traffic, there should be a good estimation of how much NTP traffic will be served (or the exact number, if using a Bandwidth Broker interacting with applications). The EF configuration proposed *must* be done by a system administrator (either a person or a software) by adding to the reserved EF bandwidth of a link the bandwidth that will be used by NTP. This necessary bandwidth will depend on the configurations of NTP associations, where the standard behavior is to use 76-byte packets (or 96-byte packets with authentication) every 1024 seconds, giving approximately 0.6 bps (or 0.75 bps with authentication) per association.

However, the average bandwidth will not be equal to the peak bandwidth, since routers will not receive uniformly sparse requests. However, practice shows that on reasonably loaded NTP servers traffic peaks are rarely more than twice the average. Therefore, system administrators *must* reserve twice the average bandwidth for NTP traffic. If the way the EF PHB is implemented at a router can starve other traffic [9], the router *must* also use a token bucket (or a functionally equivalent mechanism) to limit the rate of EF at the reserved bandwidth. If the DS Domain does not have a broker, periodic measurements on Edge Routers (connected to other DS Domains or to customer networks) can show the necessity of NTP traffic and should estimate future necessities.

EF packets carrying NTP traffic can have their DSCP remarked to a different

DSCP when entering another domain [9], as long as it is treated by a PHB that satisfies the EF PHB specification. In case of tunneling of EF packets with NTP traffic, the encapsulating packet must also be marked as EF.

4.1.2. The Hot-Potato Forwarding PHB

This work introduces The Hot-Potato Forwarding (HPF) PHB, intended to be the ideal PHB to treat packets with strong delay requirements and without delivery requirements.

The Hot-Potato Forwarding PHB can be used to construct a lossy, very low latency, very low jitter service through DS Domains. It is like a person receiving a hot potato on his or her hands: in order not to get the hands burnt, he or she cannot keep holding it and should immediately pass it on to the next person. Otherwise, he or she should simply drop it. Applications that need to send time-sensitive information, like the exchange of high-resolution timestamps (NTP) or the monitoring/metering of physical links delay (with the philosophy that monitoring should not hurt operation and losses are acceptable) can make good use of such a service. HPF suits these applications better than EF due to the limited predictability of delays in the latter. And with the loose bandwidth requirements of HPF, it will not disturb EF traffic.

It is not recommended that a DiffServ router serves a significant amount of HPF traffic. The general idea is that HPF traffic should not utilize more than 0.1% of a link, except on very special cases (e.g., a link to a public time server or an administrator-driven metering of a possibly problematic link). Since EF PHB has very strict rules, the implementation of HPF must always be done with care, in order not to break the guarantees of EF.

4.1.2.1. Definition of the HPF PHB

The HPF PHB provides forwarding treatment for a particular DiffServ aggregate called HPF traffic. HPF packets arriving at any Diffserv router *should* be immediately forwarded or dropped. No HPF packet should wait longer than a packet time (the time to send a packet in the outgoing network interface) to be sent. This assures a limited minimum jitter, based on the number of nodes and on the packet serialization delays.

No bandwidth reservations are necessary or even desired. But a maximum bandwidth is important so that misuse of HPF will not affect other PHBs, notably EF. The HPF traffic *should not* break guarantees of other PHBs. The decision of discarding or scheduling an HPF packet must be based on the guarantees of other traffic. A DiffServ router implementing HPF *should not* use traffic shaping of HPF traffic other than discarding nonconforming packets.

4.1.2.2. Implementation of the HPF PHB

Some types of scheduling mechanisms can be used to deliver a forwarding behavior similar to the one described in the previous section. Two mechanisms have shown to be capable of providing the necessary means to accomplish all the required guarantees within a proper DS Domain: *priority queue* (PQ) and *class based queuing* (CBQ).

With PQ, the HPF traffic can be served by a separate queue with the highest priority among all queues. If the queue size is measured in packets (it may be different

depending on router architecture), it should have the size of one packet. If its size is measured in bytes, it should have the size of the biggest packet it is supposed to accept. In order to prevent HPF from starving EF traffic, a token bucket *must* be used to limit HPF rate and two consecutive HPF packets should not be sent when an EF is waiting.

Using CBQ, there are many possibilities of configuration. A recommended one is to use a queue for HPF (with a mechanism to discard out-of-profile traffic, e.g., a token bucket) and a queue for EF, where the HPF queue will have a higher priority and be sized to a packet (if the queue is measured in packets) or to the size of the biggest packet expected (if measured in bytes). Again, if an HPF packet arrives when another one is being sent and there's an EF packet waiting, the HPF should be discarded. Other recommended possible implementation is to use a high priority class (no other class should have a higher priority) with two subclasses controlled using *weighted round robin* (WRR): one for HPF (a single-packet queue) and another for EF, where both HPF and EF have the same weight and a maximum transmission of one packet per turn (to avoid waiting longer than a packet to transmit). This WRR must be non-work conserving, i.e., if a queue has no packets on its turn, the other queue can transmit.

When the HPF PHB is used on a DS Domain, the *recommended* DSCP is *010011* (binary), which is in a *local use* DSCP range [18]. No efforts are made towards compatibility with legacy TOS field markings [21], except the recommended [18] compatibility with the IP Precedence [21], where *010xxx* represents immediate service.

HPF can be employed on any DiffServ router that implements any other PHBs, as long as the HPF specification is respected. If two DS Domains exchange HPF traffic, they should agree on the DSCP for HPF or should do packet remarking (mapping into an equivalent PHB, according to the HPF definition). If HPF packets are tunneled, the encapsulating packet must be also marked as HPF. Even though no reliability guarantees are made, a router should only discard an HPF packet if it really has to, in order to avoid starvation of HPF. Though no bandwidth is reserved, the fact that HPF must have the highest priority makes it interfere with the jitter observed by other traffic, especially BE.

4.2. Network Time Protocol Configuration

Some relevant issues must be discussed to gain insight into the aspects that influence clock synchronization. Some applications have strong requirements of time precision and the configuration of their synchronization infrastructure must attain to each detail. Any serious time-keeping purposes will require local network dedicated time servers (running NTP) that will synchronize from remote computers and serve time to local ones. Using a computer for tasks besides time keeping affects significantly the quality of the time served. Since NTP was designed for multitasking processors, giving it a higher priority can improve the efficiency of the clock discipline algorithms and even lower network jitter (the arrival and treatment of NTP packets will preempt other tasks). Here, equipment that introduces considerable network jitter (e.g., a hub or a radio antenna) should be avoided, especially between local time servers and the remote ones (where the presence of many intermediary equipments raises the values of the delay and the jitter).

Special care should be taken to use servers whose path to the client shares as few internet links as possible, e.g., it is recommended to use spare links whenever available, because if one intermediary link fails then more than one server will be unreachable.

4.2.1. Using the Expedited Forwarding PHB

Using too many servers for synchronizing is not a good option. NTP will always synchronize to only a few of them and will only switch servers if the quality of the data coming from the selected ones gets worse than the others (which is not common). Therefore, this work **recommends** the configuration of 3 servers for an NTP local server using the EF PHB. Only one good server is necessary on normal operation, but this redundancy is necessary for safety reasons. If the network has more than one connection to the Internet, at least one of the three NTP associations should use it.

The use of three servers will reserve about 6 times the bandwidth of a single association to a server. It will make a bandwidth reservation of 3.6 bps for EF traffic, being 1.2 bps for each path to a remote server.

4.2.2. Using the Hot-Potato Forwarding PHB

Though no bandwidth is reserved with the use of the HPF PHB, the same restrictions on the number of servers apply here. Using more than three servers is unnecessary. This work **recommends** the configuration of 3 servers for an NTP local server using the HPF PHB. Even though NTP behaves nicely in the face of high packet loss rates and the fact that HPF packets will improve the quality of the synchronization to its best (due to the assured low jitter), the remote possibility that the machine will stay hours without hearing from any server should be avoided. We also recommend that servers should be chosen carefully, from different nearby networks (i.e., with low delay to reach).

An estimate of the maximum use of bandwidth is necessary in case a network manager configures the routers along the paths of HPF traffic. If a BB is used, this information will be accurate enough and supplied by the application to the BB. Packet markings to the correct value corresponding to the HPF PHB must be done by the application itself or by the Edge Router connected to the customer network.

5. Case-Study: Evaluation of QoSYNC through Network Emulation

In order to validate the QoSYNC framework, we decided to use a network emulation environment to check how much it improves clock synchronization quality with NTP.

5.1. Network Emulation and Infrastructure

Three evaluation approaches were possible: simulation, live network measurement and network emulation.

A simulation of the specified scenario would be cheaper and runs faster than real traffic. Its tests would be controlled and reproducible. But the studied applications and the necessary traffic controlling mechanisms had already been implemented, and implementing them in a simulator would consume a lot more time than utilizing present implementations and could become a poor representation of the real environment.

The measurement of live traffic in present networks would yield trustable results and reuses the already deployed codes, but the results would not be reproducible and their analysis would be much more difficult.

The chosen method was network emulation. This method allows one to run "real" code in a controllable and reproducible environment [6], so the tests can interact with live environments as much as possible.

5.2. Testing Environment

Many elements were needed to build the necessary infrastructure for the tests. It was important to duplicate the real Internet conditions as much as possible, and the following software tools were needed for this purpose:

We used the latest stable version of the public implementation of the Network Time Protocol, NTP 4.1.1b (last updated in October 2002), with most of the configurations set as default. All machines on this experiment used Linux with kernel 2.4.18 (except the stratum 2 time-server, with FreeBSD). The kernels were recompiled to allow the use of the traffic generator, advanced routing capabilities (Diffserv) and the network emulator. We also used a tool to configure the advanced routing capabilities [8] required by Diffserv. Traffic Control, a utility of the *iproute2* package [24], was used.

A network emulation tool that would behave as a network cloud or a network link was necessary. *NIST.Net* [6] was chosen due to its simplicity, robustness and good parameterization. More details on network emulation and NIST.Net are found in [19].

Traffic Generator 2.0 (TG2) [13] was used for traffic generation, and was modified to include provisions for the use of a Pareto distribution for self-similar traffic.

We needed a monitoring tool to gather all the data during the experiment and keep it handy. RRDTool [20] was used to generate fixed-size databases, do some of the monitoring and create the simpler statistical graphs. The R platform [24] was used to compare the results and create the other graphs.

A network emulation testbed was also needed to run all this. The *NGN Testbed* from the QoSWARE Project [22] was kindly conceded for this experiment. All machines used the Mersenne-Twister [12] pseudo-random number generator. No machine on this test needed to generate more than 4 billion random numbers for all the tests, which is several orders of magnitude smaller then the period of the generator used.

5.3. Network Topology

This work needed a network topology with certain characteristics to properly analyze the proposed alterations for clock synchronization infrastructure. Two NTP machines (a client and a server) were needed. The server had to be synchronized to international standards and could be connected to an atomic physical clock.

A simple Diffserv network should separate the client and the server, with two routers connecting them. The routers should be connected to each other through a single link with 40ms delay. NTP traffic between client and server should share the link with other Internet flows (from many other networks connected to each router), which should yield common Internet conditions (packet loss and jitter). Grouping each NTP machine and the networks with their respective routers forms a network topology known as dumbbell, as shown in Figure 1. Here, the applications mark their packets with the correct DSCP. No bandwidth brokers or Edge Routers are studied as well.

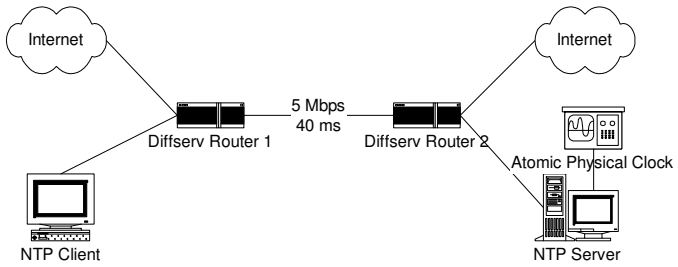

Figure 1: Studied Network Topology

This case study used a different network topology to emulate the intended topology. On the implemented topology, a few items were added or used to replace others (see Figure 2). In replacement to the external networks, each router had one machine connected to it to act as a traffic generator and as a traffic receiver. A dedicated link between NTP client and server was added to accurately measure the offset between their clocks (this would not be possible on a regular network with geographically distant machines). A network emulation machine was used in replacement of the link between the routers. The emulator delayed all packets passing in any direction by exactly 40ms, with no artificial jitter introduction. The observed jitter resulted only from queuing (verified with and without the traffic generators running).

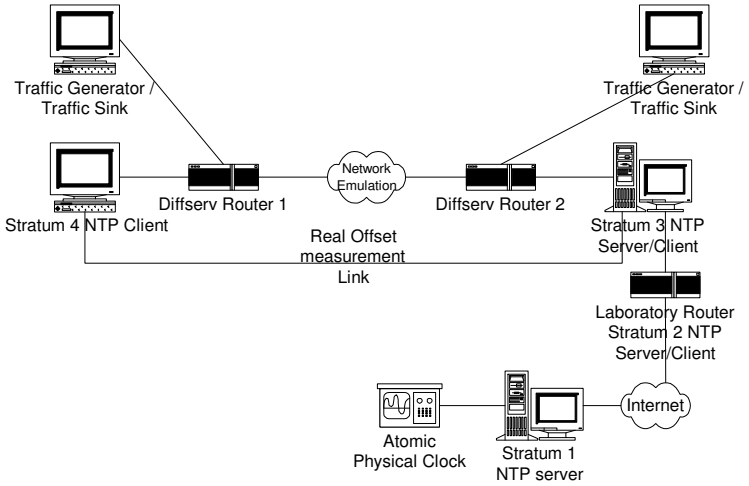

Figure 2: Implemented Network Topology

Without a physical clock, an external time reference was used, from the Internet. The laboratory router was used as a NTP gateway because local machines used private addresses. It used a remote stratum 1 NTP server and the server of the experiment (now stratum 3) used the router to synchronize. To make sure the stratum 3 server furnished a trustable time, the stratum 2 and 3 servers were configured with a short polling interval ($P = 32$ seconds) to the stratum 1 and 2 servers, respectively, and begun running several days before the start of the experiment (in order to let local clock corrections stabilize). The marking of the DS Field by the application required a code recompilation of NTP between sets of test. A dedicates link was necessary to compute real clock offsets.

It was proved that the Internet traffic is best described as having a *fractal* or *self-similar* behavior [11]. A metric was established to measure the level of self-similarity on

network traffic, called the *Hurst* parameter (H). It ranges from 0.5 (no self-similarity) to 1.0 (complete self-similarity). It is important to notice that the higher H is, the *burstier* the traffic is, which is bad for efficient network utilization (greater queuing, greater average delay, smaller average throughput). It has been shown [25] that any level of self-similarity can be achieved through the aggregation of on-off sources using a Pareto distribution to regulate it. This work used 20 UDP sources (on each generator, with packets destined to the other generator) for a peak rate of 7.2Mbps (each way, in case all sources from one side transmit at the same time) and an average generation of 3.6Mbps. It was configured to result in a significant, but common (i.e., normally observed in internet traffic studies) level of self-similarity, measured as a Hurst parameter of $H = 8.5$. In the EF scenario, one of the sources transmits EF traffic (in each generator).

5.4. Results

A preliminary test was run to decide how many replications were necessary to achieve statistical guarantees [10]. The decision was based on the maximum error acceptable for each test, calculated as a percentage of the average. The confidence level used for this analysis was 90%. Table 1 compares the results for each variable in each scenario.

Table 1: Determining Number of Replications

Analyzed Variable	Type of Traffic (scenario)	Sampled Average	Maximum Error	Number of Replications
Real Offset	Best Effort	0.009817	15%	117.8108
Real Offset	Expedited Forwarding	0.004889	15%	49.98287
Real Offset	Hot-Potato Forwarding	0.003376	15%	32.98568
Offset Estimation Error	Best Effort	0.002501	15%	110.1464
Offset Estimation Error	Expedited Forwarding	0.002707	15%	42.46485
Offset Estimation Error	Hot-Potato Forwarding	0.002421	15%	33.31118

As a result of the comparison, the test was made with 120 replications for each scenario, which resulted in 5 days (120 hours) running each of the three scenarios. Though we gathered the real and the estimated offset, the latter is only relevant on the light of the real offset. Therefore, this work analyzed a new variable, the offset estimation error, calculated as the difference between the estimated and the real offsets.

During the scenario of NTP traffic as best effort, the client lost synchrony with the server 12 times during the experiment. As NTP continues disciplining the local clock in these cases, it did not wander uncontrolled. Since this is a common event on production networks, it was considered normal operation and the samples were normally accepted for the experiment. No synchronization loss occurred on the other scenarios.

The maximum jitter was measured in each scenario for the type of traffic NTP was using with simple “ping” (with an option to change the DSField). After the run of each experiment, 1000 ping packets were sent and the difference between the maximum and minimum observed RTT (round-trip time) delay was taken and divided by two to represent the one-way delay. This difference is an estimate of the maximum possible jitter that can be observed by a traffic flow, though the jitter itself is the difference observed between two adjacent packets of a microflow. Results are shown on Table 2.

Table 2: Maximum Possible Jitter Estimate

Type of Traffic	Observed Jitter
Best Effort	1.5 ms
Expedited Forwarding	0.275 ms
Hot-Potato Forwarding	0.095 ms

The jitter observed on HPF traffic is still a little higher than what it should be, but it is believed that this was due to the use of general-purpose computers (where network activities are treated by software) to act as routers. In all types of traffic, the jitter is limited to the fact that only two routers were used, and on a given network topology the number of routers and the configuration of the queues (especially BE queue, with really varying configuration) affect the observed jitter. The three scenarios did not show any significant change in the loss rate of the background traffic, and the number of total lost packet did not differ significantly (approximately 0.499% for all tests). This was due to the use of UDP traffic, which does not use any congestion control mechanism. The goal of the background traffic was to simply imitate general Internet behavior for NTP, not to analyze how NTP would affect other types of traffic.

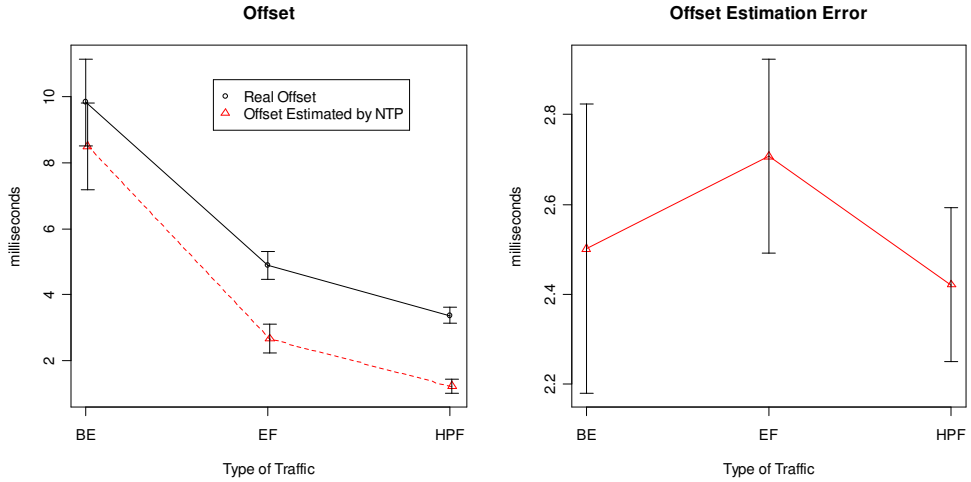

Figure 3: Offset (a) and Offset Estimation Error (b) Comparison

On Figure 3 (a) we can see a comparison between the resulting offsets between client and server of each type of traffic, computed as absolute values. The vertical bars represent the standard deviation of the data weighted by the trust interval coefficient [10], computed based on the confidence level of 90%. Undoubtedly, NTP performs systematically better as EF or HPF traffic than it does as BE traffic. As expected, HPF traffic represents a little improvement over EF traffic. The same figure shows the mean estimate of the offset. As this only looks meaningful relatively to the real offset, we must show another graphic that presents the average of the difference between the real offset and its estimate. Figure 3 (b) shows this difference, computed as absolute values.

Here, it is clear that changing the type of traffic did not significantly affect its offset estimate. Though the average has increased a little for EF, its variation is smaller, resulting in a very small aggravation of this estimate. Even though the jitter caused by EF traffic is small, and therefore the variation of the estimate represented by its vertical bars is smaller, this jitter is more unpredictable, since some packets receive a minimum

delay and some others receive a higher one. HPF traffic, however, showed a little improvement on both mean and variation of the estimation.

6. Conclusions and Future Work

This work has focused the distribution of time for clock synchronization in networked computers and proposed some network configurations based on the use of Quality of Service for dealing with it, including a novel treatment of packet forwarding.

It was shown that the use of the Expedited Forwarding PHB or the proposed Hot-Potato Forwarding PHB can help ensure permanent synchronization of a computer with NTP and that it can also improve the quality of such synchronization. There was no need to alter the behavior of NTP, with the exception of QoS markings.

Many networks can benefit from the use of this proposal, which can decrease the overall use of NTP traffic on the Internet (it is well known that in order to recover from its synchronization losses NTP raises its bandwidth) and improve its accuracy.

We devised two interesting future works derived from this work. The first is a broader study of the peak-to-mean relation for NTP traffic. With it, one can make a better estimation of the necessary values for peak bandwidth reservation when using EF traffic. The second is a study of the effect of the use of HPF traffic on TCP connections (of course, not using HPF *for* TCP). While HPF should not affect TCP loss rate significantly, it is expected to affect TCP traffic jitter. Both require further investigation.

7. Acknoledgements

The authors would like to thank CAPES for partially funding this work, and Prof. Dr. Djamel Fawzi Hadj Sadok for all the help and support.

8. References

[1] AWDUCHE, D. et al, “Requirements for Traffic Engineering over MPLS”, RFC 2702, September 1999.

[2] BARBOSA, Rodrigo et al. “IPstat: Uma Ferramenta para Medições Unidirecionais na Internet”, submitted to Salão de Ferramentas in SBRC2004, 2004.

[3] BLAKE, S., BLACK, D. L., CARLSON, M., DAVIES, E., WANG, Z. et WEISS, W., “An Architecture for Differentiated Services”, RFC 2475, December 1998.

[4] BRADEN, R., CLARK, D. D., et SHENKER, S., “Integrated Services in the Internet Architecture: an Overview”, RFC 1633, June 1994.

[5] CALLADO, Arthur C. “Clock Synchronization in Computer Networks with Quality of Service”, master in science dissertation, Universidade Federal de Pernambuco, September 2003.

[6] CARSON, Mark, “Application and Protocol Testing through Network Emulation”, Internetworking Technologies Group, September 1997.

[7] HEINANEN, J., BAKER, F., WEISS, W. et WROCLAWSKI, J.. “Assured Forwarding PHB Group”, RFC 2597, June 1999.

[8] HUBERT, Bert et al, “Linux Advanced Routing and Traffic Control HOWTO”, http://lartc.org, December 2002.

[9] JACOBSON, V., NICHOLS, K. et PODURI, K. “An Expedited Forwarding PHB”, RFC 2598, June 1999.

[10] JAIN, Raj. “The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling”. John Wiley & Sons. 1991.

[11] LELAND, Will E. et al. “On the Self-Similar Nature of Ethernet Traffic (extended version)”. IEEE/ACM Trans. on Net., volume 2, number 1. February 1994.

[12] MATSUMOTO, Makoto; NISHIMURA, Takuji. “Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator”. ACM Transactions on Modeling and Computer Simulations: Special Issue on Uniform Random Number Generation. 1998.

[13] MCKENNEY, P., LEE, Dan Y., DENNY, Barbara A., “Traffic Generator Software Release Notes”, SRI Int. and USC/ISI PCEN, January 2002.

[14] MILLS, David L., “Adaptive Hybrid Clock Discipline Algorithm for the Network Time Protocol”, IEEE Trans. on Net., volume 6, number 5, October 1998.

[15] MILLS, David L., “Experiments in Network Clock Synchronization”, RFC 957, September 1985.

[16] MILLS, David L., “Network Time Protocol (Version 3) Specification, Implementation and Analysis”, RFC 1305, March 1992.

[17] MILLS, David L., “Simple Network Time Protocol (SNTP) Version 4 for IPv4 IPv6 and OSI”, RFC 2030, October 1996.

[18] NICHOLS, K. et al, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”, RFC 2474, December 1998.

[19] NIST, “NIST Net Home Page”, http://dns.antd.nist.gov/itg/nistnet/.

[20] OETIKER, Tobias, “RRDTool Manual”, http://www.rrdtool.com, November 2002.

[21] POSTEL, J. “Internet Protocol”, RFC 791, September 1981.

[22] QoSWARE, “Gerenciamento de QoS no MIDDLEWARE para Aplicações em Tempo-Real”, http://www.cin.ufpe.br/~gprt/qosware, November 2001.

[23] RNP - Projeto NTP, “Lista dos servidores NTP Stratum 2 no Brasil”, http://www.rnp.br/cais/ntp/ntp_stratum2.html.

[24] R-Project, “The R Project for Statistical Computing”, http://www.r-project.org.

[25] WILLINGER, Walter et al. “Self-Similarity Through High-Variability: Statistical Analysis of Ethernet LAN Traffic at the Source Level”. IEEE/ACM Transactions on Networking, volume 5, pp. 71-86. February 1997.

[26] XIAO, Xipeng et NI, Lionel, “Internet QoS: A Big Picure”, IEEE Network, March 1999.