the Availability Digest

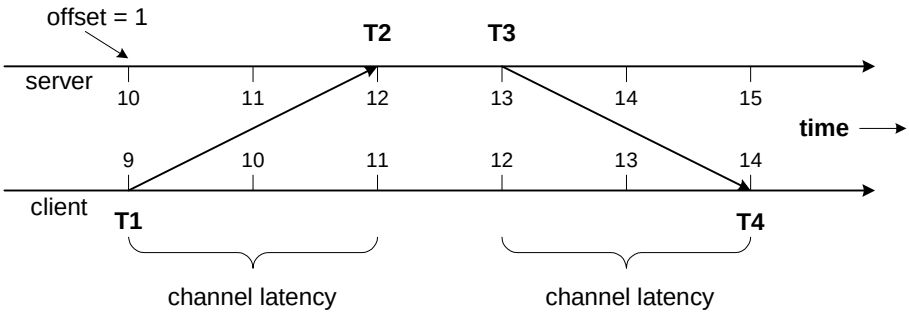
Time Synchronization for Distributed Systems – Part 2

December 2007

Distributed systems often require that their nodes and the clients that access them all have the same view of time. In Part 1 of this three-part series, we showed how NTP calculates the time offset of a client relative to a time server. However, coordinating a client with a single time server leaves a lot of room for error. Here in Part 2, we describe the NTP facilities that allow us to reduce this error significantly.

NTP Time Offset Determination – A Review

Reviewing Part 1, we showed how a client can estimate the time difference (the time offset) between itself and a time server that is presumably a more trusted keeper of time. The client does this simply by sending a timestamped message to the time server, which adds its own timestamp and returns the message to the client. This process is shown in Figure 1:



Measuring Clock Error
Figure 1

The client sends a message to the time server with a timestamp T1. Upon receipt, the time server adds its timestamp T2 to the message. It then returns the message to the client, adding its time of transmission, T3, to the message. The client notes that it receives the message at time T4. In Part 1, we showed that the time offset between the client and the time server is

$$\text{time offset} = \frac{(T2 - T1) - (T4 - T3)}{2}$$

In Figure 1, we see that the client is slow compared to the time server by one clock tick (when the time server thinks that the time is 10, the client thinks that it is 9). From the above equation, the time offset in the example of Figure 1 is [(12 – 9) – (14 – 13)] / 2 = 1, as expected. Therefore, one tick must be added to the client clock.

Time Offset Errors

One source of error in this offset calculation is channel latency jitter. Over the Internet especially, the time that it will take a message to get from its source to its destination may change rapidly and significantly as congestion comes and goes and as routing changes. Software can also cause jitter, though this is typically much less than communication jitter (microseconds rather than milliseconds). This problem of jitter was discussed in Part 1 and is solved by averaging the time offset over several measurements taken over several minutes.

Another problem is channel asymmetry. As is seen in Figure 1, it takes a certain amount of time for the client message to reach the server. We call this time the channel latency. The response to the server is also delayed by the channel latency. In Figure 1, the channel latency in either direction is the same and is two clock ticks.

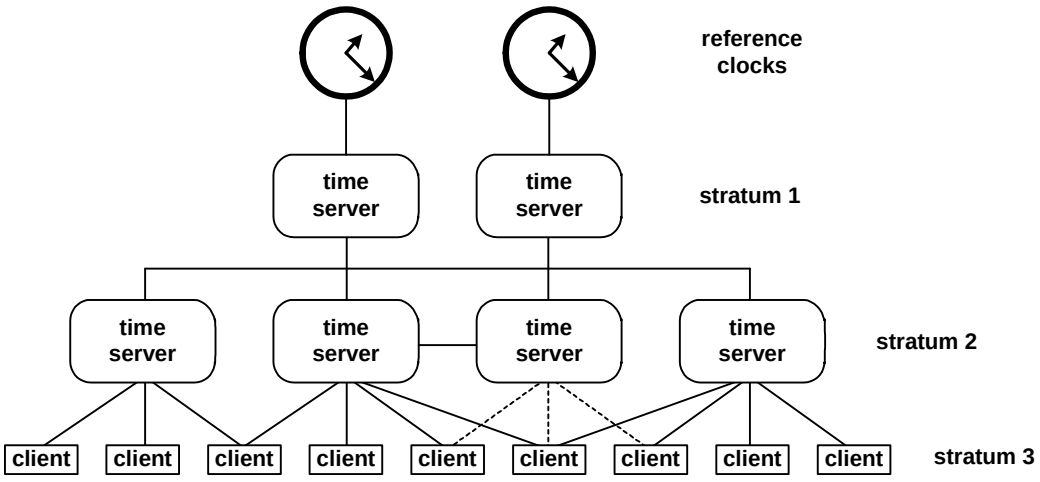
The derivation of the time offset shown previously assumes that the two channel latencies are equal. That is, the channel is symmetric. Over a dedicated line, this assumption might be valid. However, over the Internet, these latencies are very likely to be different; and perhaps they may be even significantly different. This is because the request message from the client and the response message from the time server may flow over different routes. Furthermore, the congestion over these routes may be significantly different. Even worse, routing and congestion may change over time. This represents a major source of time calculation error.

For instance, in Figure 1, if the channel latency from the client were only one tick, and if the channel latency from the time server were three clicks, T2 would be 11 and T3 would be 12. The calculated time offset would be [(11 - 9) – (14 – 12)] / 2 = 0, which is wrong.

The problems of channel asymmetry and other less significant problems are solved by averaging the time offset derived from time readings from several time servers rather than just one. To accomplish this, the time servers and their clients are arranged in an NTP subnet.

NTP Subnet Topology

An NTP subnet comprises distributed time servers operating in a self-organizing, hierarchical master/slave configuration, as shown in Figure 2.



NTP Hierarchical Organization
Figure 2

At the top of the hierarchy are the civil-time reference clocks with which we want to synchronize. These may be considered stratum 0, and may include atomic clocks, radio sources, satellite sources, and so on.

Time servers that synchronize directly with the reference clocks are said to be stratum 1 time servers. Time servers that synchronize from stratum 1 clocks are stratum 2 time servers, and so on. Though NTP allows many stratum levels, typical NTP subnets comprise only a few levels (typically four or less). This is because as the stratum level increases, the accuracy of the clock decreases.

Clearly, the lower the stratum level, the more accurate the time server. Whenever a time server (or client for that matter) enters the network, it synchronizes with some time server that is a stratum k time server. It then becomes a node at the stratum (k+1) level. Thus, the network is self-organizing.

Subnet Node Relationships

There are several different relationships that nodes may have with each other.

Peer-to-Peer

Nodes may be peers to each other at a given stratum level. Each peer node in a peer group validates its time against that of the other peers by acting as a client to the other peer time servers. In this way, the nodes in the peer group maintain a consensus among themselves as to what the correct time is. Each member of the peer node may also adjust its time by communicating with a time server in the next lower stratum.

Only one of the peers acts as a time server. The peers can decide which among them has the most accurate time, and that peer becomes the time server. Thus, the actual physical time server in a peer group can change from time to time.

Client/Server

In a client server, or master/slave, relationship, a client node at one stratum level will adjust its time to be that of a time server at the next lower stratum level. The client may do so with a number of time servers at that stratum level and average the results to achieve better accuracy, as described later.

Broadcast/Multicast

In a subnet characterized by many clients, network traffic can be minimized by having a time server broadcast or multicast time messages to many clients (similar to a radio broadcast to stratum1 time servers). The clients add measured or preconfigured channel latency times to the time contained in the message from the time server to determine their own clock settings. Broadcast/multicast does not provide the accuracy that is obtained by client/server interactions, but it is more efficient in terms of network traffic and time server load.

Multiple Roles

As can be seen from Figure 2, any particular physical system can play one or more of these roles. A system may be a client to one or more time servers. It can act as a peer to other time servers in a peer group. It can also act as a time server to other clients.

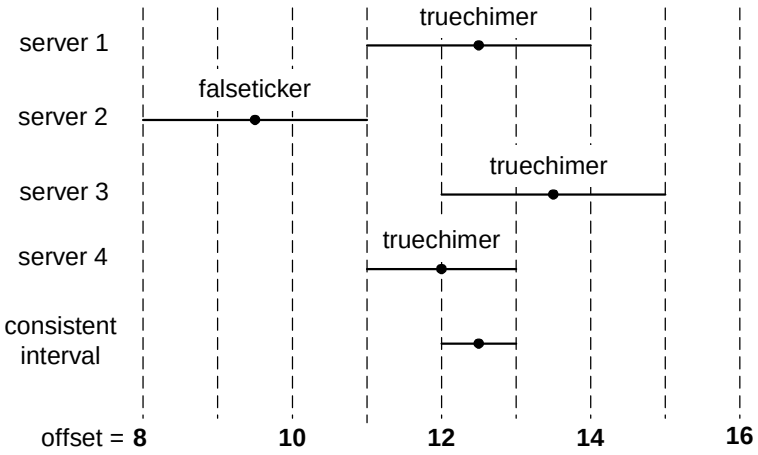
Combining Clocks

The effects of errors due to jitter, channel asymmetry, and other causes can be smoothed out by taking readings from several time servers and averaging them. NTP does not treat this averaging casually. There are two steps to this process – discarding clocks whose times are inconsistent with the group and then weighting the remaining offsets according to the perceived accuracies of their clocks.

Clock Selection - Truechimers and Falsetickers

The first step taken by a node that uses clock-combining is to send a time message to each of the time servers that it is using and then to calculate the offsets from each time server. The next step is to determine the optimum set of offsets to average. This is done via the *Intersection Algorithm*, which is a refined version of Marzullo's algorithm.[1] Finally, the chosen sets of offsets are averaged to obtain the final offset estimate.

The Intersection Algorithm uses the confidence intervals of each time server to determine which clocks are out of range with the majority of clocks and thus should be ignored (the *falsetickers*) and which are reasonable to use (the *truechimers*). The confidence interval for a time server is determined from the dispersion of its measured offsets over a period of time. Figure 3 shows this process in simplified form.



The Intersection Algortihm
Figure 3

This figure shows offset measurements taken from four time servers, along with their confidence intervals. For instance, the offset measured with respect to Server 1 is 12.5 ± 1.5 (i.e., within a specified confidence level, the offset is somewhere between 11 and 14).

The intersection of these confidence intervals that includes the most servers is then chosen. In Figure 3, the interval 12 to 13 includes three servers, Server 1, Server 3, and Server 4, and is the chosen interval. All servers that include this interval are the truechimers (Servers 1, 2, and 3). Any servers that do not include this interval are falsetickers (Server 2).

Offsets from falsetickers are discarded. The offsets of truechimers are included in the averaging, described next.

[1] Marzullo, Keith, *Maintaining the Time in a Distributed System: An Example of a Loosely-Coupled Distributed Service*, Ph.D. dissertation, Stanford University, Department of Electrical Engineering; February, 1984.

Clock Averaging

A weighted average of the offsets of the truechimers is now taken. The weight of each offset is based on the confidence level of that offset. Therefore, offsets obtained from time servers that are deemed to be more reliable are given heavier weights in the resulting average.

This weighted average is used as the offset for the client's clock and is used to adjust its clock. Over a period of time, the effects of jitter, asymmetric channels, and other random errors will be averaged out by this process.

SNTP

SNTP (Simple Network Time Protocol) is a stripped down version of NTP. It is a client-only protocol that supports only synchronizing with a single time source. Since this is a common configuration, SNTP can lead to a smaller footprint in the client system and is easier to manage. The downside is that it does not provide the accuracy of NTP since it cannot average over multiple time servers. One second accuracies are typical.

SNTP is specified by RFC 1361.

Security

One final point is to be made. If time adjustment messages are sent in the clear, the NTP subnet is susceptible to attack. For instance, an attacker could intercept time responses and modify the time stamps in the message before passing them on to the client. Great havoc can be wreaked by forcing the subnet nodes to show drastically differing times.

To thwart this, NTP provides a security function that encodes the time messages so that they cannot be hacked. The recommended encryption method uses symmetric key cryptography.

What's Next?

In these first two parts on time synchronization in distributed systems, we have described what today's civil time really is and how NTP synchronizes a network of distributed systems to that time.

However, there are cases in which the nodes in a system must be kept in time synchronism but not necessarily in synchronism with real time. In these systems, the nodes must agree on time for proper event ordering; but if the time observed by these systems is a few seconds off, there is no penalty. In Part 3, we describe logical clocks first proposed by Leslie Lamport in a seminal paper that accomplishes this task in a way that is much simpler than NTP.