The ISP Column
A monthly column on things Internet

March 2026
Geoff Huston

Revisting Time

On the last day of December 2016 there was a brief hiccup as the world's clocks adjusted their time according to the *Coordinated Universal Time* (UTC) standard by adding an extra second to the last minute of the 31st of December. This is not unusual by any means, and this is the 27th time UTC has been adjusted by the insertion of this *leap second* into the UTC timestream.

In past years this has been the cause of some major IT disasters, as many applications don't expect a minute to have 61 seconds, and applications have crashed or spun into an infinite loop as a result. The issue was that leap seconds happen infrequently and code that could recognise the occurrence of a leap second and accommodate it was rarely written and never tested in production! Exacerbating the issue was that these problems could appear at the stroke of midnight on New Year's Eve UTC, hardly a time where a sober IT response team can be assembled at short notice!

Over the last decade or so we've improved our act. The code base is probably as bad as ever, but now the time synchronisation protocol can be used to "smear" a single one-second time leap into a thousand millisecond corrections, spread out over many hours.

So far, this leap second smearing approach has been very effective in avoiding global IT disasters at the times of leap second insertions.

The reason behind *leap seconds* has been the combination of increasingly accurate clocks and a wobbly Earth.

Our definition of time is based on the regular rotation of the earth about its own axis, where one period of rotation is 86,400 seconds. In other words, a *second* is defined as precisely 1/86,400 of the time for the Earth to rotate precisely once around its own axis. We've progressively refined our ability to measure time, from water clocks to pendulums, to mechanical escapement mechanisms and high precision mechanical clockwork to quartz oscillators and then to atomic transitions. Today's standard definition of a *second* is *Atomic Time* (TAI), which is 9,192,631,770 periods of the radiation emitted by a caesium-133 atom in the transition between the two hyperfine levels of its ground state (termed a *SI second*).

The Earth is nowhere near as stable when used as a reference clock. The tidal forces exerted on the earth by the moon tend to slow down the rotational speed of the Earth by some 2.3ms per century, while the movement of mass on the earth's surface, such as the seasonal variation of polar icecaps, glacial variation, or oceanic earthquakes, act to both speed up and retard the Earth's rotational velocity. For the past few centuries, the average period of the Earth's rotation has been slowing by some 1.4 – 1.7ms per century (Figure 1).

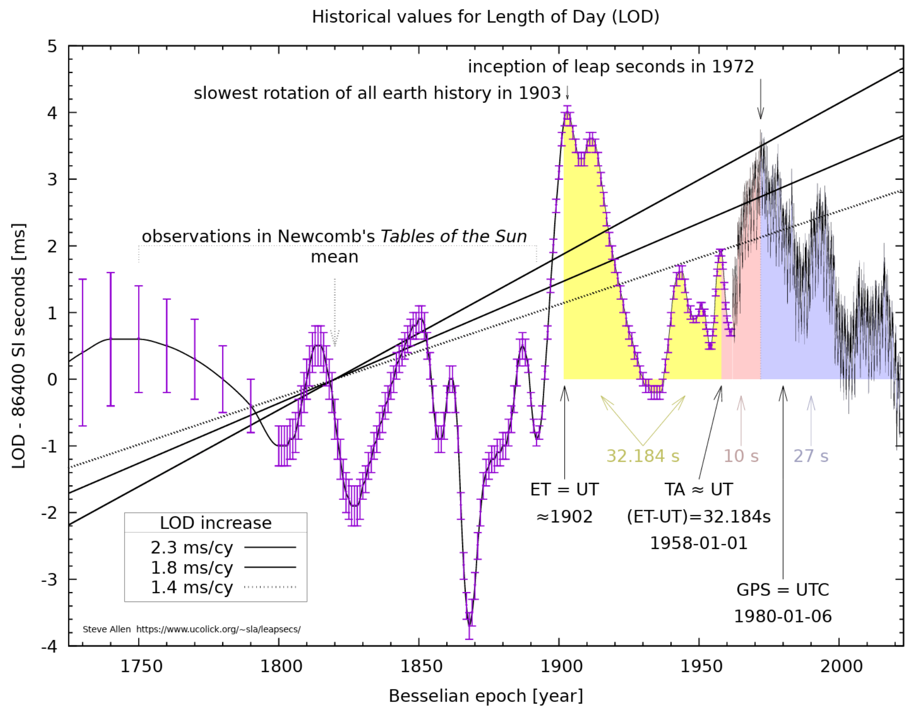

Figure 1 – Historical Values of the length of a day in units of SI seconds, Steve Allen (https://www.ucolick.org/~sla/leapsecs/historiclodmshz.pdf)

The way we've accommodated a highly regular definition of time and a somewhat irregular planetary rotation has been to allow these time systems to drift apart up to a threshold value, and then insert a full second into the UTC timestream at either mid-year, or at the end of the year. The objective of these corrections is to ensure that the two systems differ by at most 0.9 seconds. The International Earth Rotation and Reference Systems Service (IERS) has been delegated the responsibility for scheduling leap seconds, releasing a bulletin six months in advance of each potential leap second scheduled time. A view of leap seconds since 1973 is shown in Figure 2.

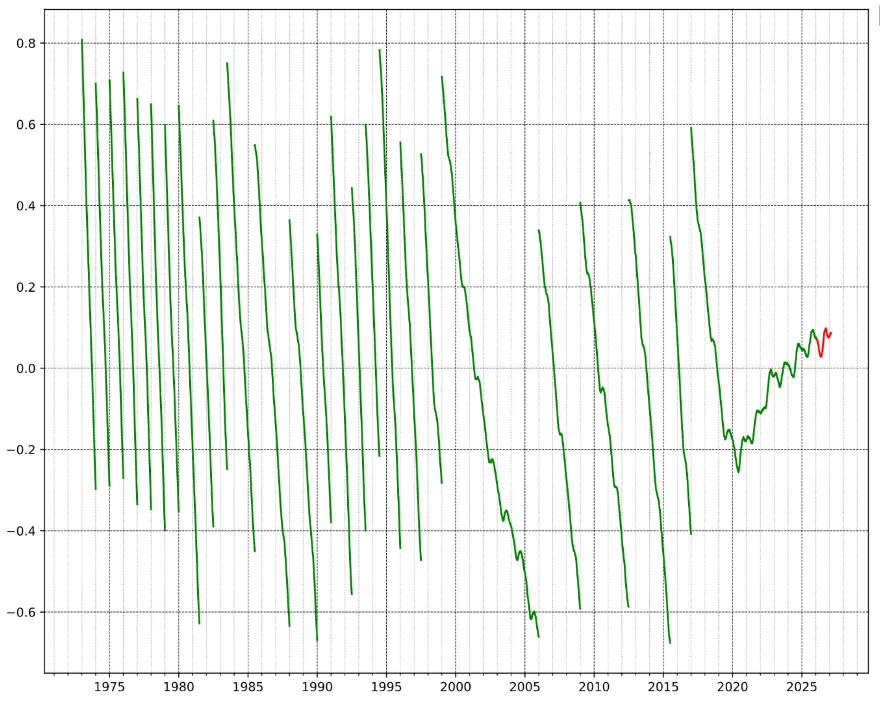

Figure 2 – Insertion of Leap Seconds applied to Coordinated Universal Time (UTC) (https://en.wikipedia.org/wiki/Leap_second#/media/File:Leapsecond.ut1-utc.svg)

The entire topic of time, time standards, and the difficulty of keeping a highly stable and regular clock standard in sync with a slightly wobbly rotating Earth has been a longstanding debate in the *International Telecommunication Union Radiocommunication Sector* (ITU-R) standards body that oversees this coordinated time standard. I am not sure that anyone would argue that the challenges of synchronizing a highly accurate reference time signal with a less than perfectly rotating planet is sufficient reason to completely discard the concept of a coordinated time standard and just let each computer system drift away on its own concept of time. Indeed, many services we use today are based on a distributed platform environment, and a highly accurate single reference time signal is essential to coordinate the actions of each individual platform with its peers. Our digital world needs a highly accurate coordinated reference time as one of its essential pre-requisites. The current argument is whether anyone cares as to how well this reference time aligns to the period of the earth's rotation, and the general consensus today appears to be that this alignment is little more than a troublesome historical notion that should be abandoned, as well as any further *leap seconds*.

These days we have become used to a world that operates on a consistent time standard, and we have become used to all of our computers operating at sub-second precision and we cannot conceive of an environment where this is not the case. But how do we coordinate time across the Internet? In this article I will look at how a consistent time standard is spread across the Internet and examine the operation of the *Network Time Protocol* (NTP), and the recent efforts to add aspect of security to this protocol to protect the integrity of time distribution.

Universal Time Standards

It would be reasonable to expect that the time is just the time, but that is not the case. The Universal Time reference standard has several versions, but there are two standards are of interest to network timekeeping.

Universal Time (UT1) is the principal form of Universal Time. Its definition is based on the period of time for the Earth to rotate about its axis with respect to the Sun. Although conceptually it is *Mean Solar Time* at 0° longitude, extremely precise measurements of the Sun are difficult. Hence, UTI is computed from observations of distant quasars using long baseline interferometry, laser ranging of the Moon and artificial satellites, as well as the determination of GPS satellite orbits. UT1 is the same everywhere on Earth and is proportional to the rotation angle of the Earth with respect to distant quasars, specifically the *International Celestial Reference Frame* (ICRF), neglecting some small adjustments.

Coordinated Universal Time (UTC) is an atomic timescale that approximates UT1. It is the international standard on which civil time is based. UTC time ticks in units of *SI seconds*, in step with *International Atomic Time* (TAI). It usually has 86,400 SI seconds per day but is kept within 0.9 seconds of UT1 by the introduction of occasional intercalary leap seconds. From 1972 when the concept of leap seconds was introduced to the present time (2026) these leap seconds have always been positive, although the process includes the capability to retard the UTC clock by a second through the insertion of a *negative leap second*.

NTP, Time, and Timekeeping

The objective of the Network Time Protocol (NTP) (RFC 5905) is simple: to allow a client computer to synchronize its clock with UTC time, and to do so with a high degree of accuracy and a high degree of stability. Within the scope of a Wide Area Network (WAN), NTP will provide an accuracy of small numbers of milliseconds. As the network scope gets finer, the accuracy of NTP can increase, allowing for sub-millisecond accuracy on LANs and sub-microsecond accuracy when using a precision time source such as a *Global Positioning System* (GPS) receiver or a caesium oscillator.

If a collection of clients all use NTP, then this set of clients can operate with a synchronized clock signal. A shared data model, where the modification time of the data is of critical importance, is one example of the use of NTP in a networked context. For example, I've relied on NTP timer accuracy at the microsecond level when trying to combine numerous discrete data sources, such as a web log on a server combined with a *Domain Name System* (DNS) query log from DNS resolvers and a packet trace.

To consider NTP, it is necessary to consider the topic of timekeeping itself. It is useful to introduce some timekeeping terms at this juncture:

Stability	How well a clock can maintain a constant frequency
Accuracy	How well the frequency and absolute value of the clock compares with a standard reference time
Precision	How well the accuracy of a clock can be maintained within a particular timekeeping system
Offset	The time difference in the absolute time of two clocks
Skew	The variation of offset over time (first-order derivative of offset over time)
Drift	The variation of skew over time (second-order derivative of offset over time)

NTP is designed to allow a computer to be aware of three critical metrics for timekeeping: the *offset* of the local clock to a selected reference clock, the *round-trip delay* of the network path between the local computer and a selected reference clock server, and the *dispersion* of the local clock, which is a measure of the maximum error of the local clock relative to the reference clock. Each of these components is maintained separately in NTP. They provide not only precision measurements of offset and delay, to allow the local clock to be adjusted to synchronize with a reference clock signal, but also definitive maximum error bounds of the synchronization process, so that the user interface can determine not only the time, but the quality of the time as well.

NTP uses UTC, as distinct from the *Greenwich Mean Time* (GMT), as the reference clock standard. UTC uses the TAI time standard, implying that, like UTC itself, NTP has to incorporate leap second adjustments from time to time.

NTP is an "absolute" time protocol, so that local time zones—and conversion of the absolute time to a calendar date and time with reference to a particular location on the Earth's surface—are not an intrinsic part of the NTP protocol. This conversion from UTC to the wall-clock time, namely the local date and time, is left to the local computer.

NTP Servers and Clients

NTP uses the concepts of *server* and *client.* A server is a source of time information, and a client is a system that is attempting to synchronize its clock to a server.

Servers can be either a *primary server* or a *secondary server.* A *primary server* (often also referred to as a *stratum 1* server using terminology originally borrowed from the time reference architecture of the telephone network) is a server that receives a UTC time signal directly from an authoritative clock source, such as a configured atomic clock or—very commonly these days—a GPS signal source. A *secondary server* receives its time signal from one or more *upstream servers,* and distributes its time signal to one of more *downstream servers* and *clients.* Secondary servers can be thought of as clock signal repeaters, and their role is to relieve the client query load from the primary servers while still being able to provide their clients with a clock signal of comparable quality to that of the primary servers. The secondary servers need to be arranged in a strict hierarchy in terms of upstream and downstream, and the stratum terminology is often used to assist in this process.

As noted previously, a stratum 1 server receives its time signal from a UTC reference source. A stratum 2 server receives its time signal from a stratum 1 server, a stratum 3 server from stratum 2 servers, and so on. A stratum *n* server can peer with many stratum $n - 1$ servers in order to maintain a reference clock signal. This stratum framework is used to avoid synchronization loops within a set of time servers.

Clients peer with servers in order to synchronize their internal clocks to the NTP time signal.

The NTP Protocol

At its most basic, the NTP protocol is a clock request transaction, where a client requests the current time from a server, passing its own time with the request. The server writes its local time to the data packet and passes the packet back to the client. When the client receives the packet, the client can derive two essential pieces of information: the reference time at the server and the elapsed time, as measured by the local clock, for a signal to pass from the client to the server and back again. Repeated iterations of this procedure allow the local client to remove the effects of network jitter and thereby gain a stable value for the delay between the local clock and the reference clock standard at the server. This value can then be used to adjust the local clock so that it is synchronized with the server. Further iterations of this protocol exchange can allow the local client to continuously correct the local clock to address local clock skew.

NTP operates over the *User Datagram Protocol* (UDP) with an unencrypted payload. An NTP server listens for client NTP packets on UDP Port 123. The NTP server is stateless and responds to each received client NTP packet in a simple transactional manner by adding fields to the received packet and passing the packet back to the original sender, without reference to preceding NTP transactions.

Upon receipt of a NTP packet from a client, the server time-stamps receipt of the packet as soon as possible within the packet assembly logic of the server. The packet is then passed to the NTP server process. This process interchanges the IP Header Address and Port fields in the packet, overwrites the reference timestamp field in the NTP packet with local clock value, time stamps the egress of the packet, recalculates the checksum, and sends the packet back to the client.

The NTP packets sent by the client to the server and the responses from the server to the client use a common format, as shown in Figure 3.

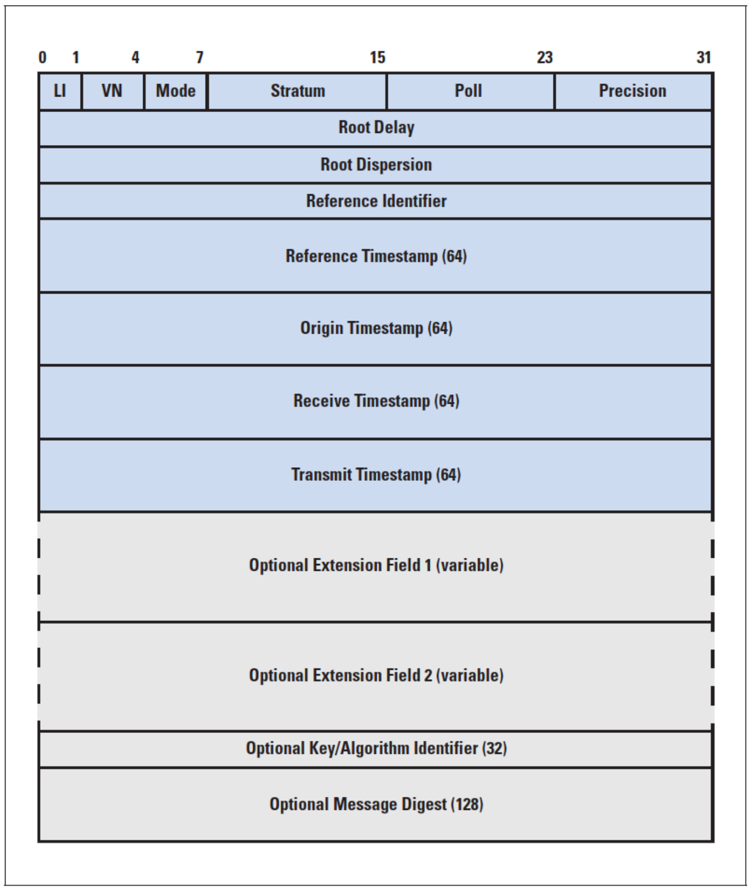

Figure 3: NTP Message Format

The basic operation of the protocol is that a client sends a packet to a server and records the time the packet left the client in the *Origin Timestamp* field (T1) of the NTP packet. The server records the time the packet was received, *Receive Timestamp* (T2). A response packet is then assembled with the *Transmit Timestamp* is set to the time that the message is passed back toward the client (T3). The client then records the time the packet arrived (T4).

Upon receiving an NTP response, the client can estimate the delay between the client and the server. The transmission delay can be calculated as the total time from transmission of the poll to reception of the response minus the recorded time for the server to process the poll and generate a response:

$$\delta = (T4 - T1) - (T3 - T2)$$

The offset of the client clock from the server clock can also be estimated by the following:

$$\Theta = ½ [(T2 - T1) + (T3 - T4)]$$

It should be noted that this calculation assumes that the network path delay from the client to the server is the same as the path delay from the server to the client, is that the paths between client and server are symmetric.

NTP uses the minimum of the last eight delay measurements as δ_0. The selected offset, Θ_0, is one measured at the lowest delay. The values (Θ_0,δ_0) become the NTP update value.

When a client is configured with a single server, the client clock is adjusted by a slew operation to bring the offset with the server clock to zero, as long as the server offset value is within an acceptable range.

When a client is configured with numerous servers, the client will use a selection algorithm to select the preferred server to synchronize against from among the candidate servers. Clustering of the time signals is performed to reject outlier servers, and then the algorithm selects the server with the lowest stratum with minimal offset and jitter values.

When NTP is configured on a client, it attempts to keep the client clock synchronized against the reference time standard. To do this task NTP conventionally adjusts the local time by small offsets (larger offsets may cause side effects on running applications, as has been found when processing leap seconds). This small adjustment is undertaken by an *adjtime()* system call, which slews the clock by altering the frequency of the software clock until the time correction is achieved. Slewing the clock is a slow process for large time offsets; a typical slew rate is 0.5 ms per second.

Obviously, this informal description has taken a rather complex algorithm and some rather detailed math formulas without addressing the details.

Securing NTP

NTP operates in the clear, and it is often the case that the servers used by a client are not local. This provides an opportunity for an adversary to disrupt an NTP session, by masquerading as a NTP server, or altering NTP payloads in an effort to disrupt a client's time-of-day clock. Many application-level protocols are time sensitive, including TLS, HTTPS, DNSSEC and NFS. Most Cloud applications rely on a coordinated time to determine the most recent version of a data object. Disrupting time can cause significant chaos in distributed network environments.

While it can be relatively straightforward to secure a TCP-based protocol by adding an initial TLS handshake and operating a TLS shim between TCP and the application traffic, it's not so straightforward to use TLS in place of a UDP-based protocol for NTP. TLS can add significant jitter to the packet exchange. Where the privacy of the UDP payload is essential, then DTLS might conceivably be considered, but in the case of NTP the privacy of the timestamps is not essential, but the veracity and authenticity of the server is important.

NTS, a secured version of NTP, is designed to address this requirement relating to the veracity and authenticity of packets passed from a NTS server to an NTS client. The protocol adds a NTS Key Establishment protocol (NTS-KE) in additional to a conventional NTPv4 UDP packet exchange (RFC 8915).

An NTS client connects to an NTS-KE server using TLS. Once the TLS session is established, including the client's authentication of the NTS-KE server as part of the TLS session establishment, the parties then negotiate some additional protocol parameters. The server sends the client a supply of cookies, and an address and port of the server's NTP server for which the cookies are valid. The parties use TLS key export (RFC 5705) to extract key material, which will be used in the next phase of the protocol. This negotiation takes only a single exchange once the TLS handshake is completed, after which the server closes the connection and discards all associated TLS and TCP state. At this point, the NTS-KE phase of the protocol is complete. Ideally, the client should not need to connect to the NTS-KE server again.

The client then proceeds to exchange NTPv4 packets with the nominated NTS server. The client sends the server an NTP client packet that includes several extension fields. Included among these fields is a cookie (previously provided by the NTS Key Establishment server) and an authentication tag, computed using key material extracted from the NTS-KE handshake. The NTS server uses the cookie to recover the key material from the cookie and send back an authenticated response. The response includes a fresh, encrypted cookie that the client then sends back in the clear in a subsequent request. The role of cookies in NTS is analogous to that of TLS session tickets (RFC 5077). The NTS server does not hold and per-client state, as all state is held on the client side and sent to the NTS server through these cookies.

The extension fields define a collection of NTS fields for cryptographically securing NTPv4 using previously established key material. They are suitable for securing client-server mode because the server can implement them without retaining per-client state. All state is kept by the client and provided to the server in the form of an encrypted cookie supplied with each request.

This is a sensible and pragmatic approach to securing the NTP protocol. It appears to borrow from the HMAC approach, allowing a NTP client to validate that the NTP message has not been tampered with while in flight, that the message comes from the trusted sender who possesses the key value that was established on session start, and prevents potential attackers from re-sending captured packets. It leaves the timer exchange essentially unaltered, so the time synchronization behaviours are the same for NTS.

NTP over QUIC

It's a recent trend in the IETF to define a migration path for almost every application that operates over TCP or UDP to use QUIC, and the NTP protocol is no exception.

There is a recent Internet draft, "Time Synchronization over QUIC" (draft-mccollum-ntp-tsq) (TSQ) that proposes precisely this. It's a relatively straightforward shift of of NTP's UDP transport protocol, allowing the choice of using QUIC's reliable stream service (similar to a TLS transport service) or QUIC's datagram mode (which can be considered as functionally close to a DTLS service).

QUIC runs all stream packets within an end-to-end encrypted envelope, and in the case of NTP it's not easy to understand what this offers over NTS. In both cases the client can authenticate the server and use a cryptographic test to ensure that the message has not been altered and is not being replayed. In the case of NTS the time values are passed in the clear, while TSQ obscures these values as well inside QUIC's encrypted payload. Quic's reliable stream mode is little different to using TLS in this context and is ill-suited to the demands of the NTP protocol exchange. With respect to using QUIC's datagram mode it's entirely unclear what is gained here, and the potential downside is that the additional packet processing encryption/decryption overhead adds to the time delay and jitter component, undermining the underlying time synchronization objective.

The TSQ draft has expired in February 2026, and it's probably better if it were not revived!

Is Time a Public or Private Service?

It has often been the case that the costs of the operation of service infrastructure for highly precise time has been borne by the public purse, largely motivated by military needs. Highly accurate time enables high precision calculation of position, and a highly accurate position service has a myriad of applications in military spheres, as well as its many civil applications.

The U.S. Global Positioning System (GPS) is utility service operated by the US Space Force that provides free, continuous worldwide positioning, navigation, and time services to both civilian and military users. It consists of 31 satellites currently in operation, and is a critical, albeit vulnerable infrastructure for global navigation and timing. The project was originally commissioned as a US military service and was subsequently opened up general use. Russia operates the GLONASS, Europe operates the Galileo system, and there is also China's BeiDou system. All of these services are perceived as national strategic assets in addition to their general utility as a time and positioning service.

These satellite-based time services are fed their time signals from a collection of atomic clocks. Other countries also use atomic clocks and use radio signals to broadcast a time service. Many of these bodies also support the operation of public NTP services. The strategic weakness with all these satellite-based systems (and related radio systems) is that they are vulnerable to jamming. The open nature of the NTP protocol presents some comparable issues in terms of vulnerability to disruption.

A number of time service operators have decided to use of NTS as the means of a more resilient distribution of a time service. Sweden, through the Swedish Post and Telecom Authority, has launched a public distributed time service that is based on a national pool of NTS-accessible stratum 1 timeservers. However, the shift to NTS for such public time services has not been so widespread, and it appears that Brazil and Germany are the two other countries that operate NTS services as a public function. In other cases, it has been left to the private sector and motivated individuals to support the NTS time distribution network. It does seem somewhat contradictory that some national communities see value in operating reference atomic clocks as a national asset, and disseminate this time signal by radio and NTP, but the step to NTS as a means of improving the resilience of this service so far has appeared to be just one step too far!

Disclaimer

Author

Geoff Huston AM, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net