

Proposal to GRID Forum for a Timestamp Standard

Author: Bodo Parady, Sun Microsystems

Email: bodo@eng.sun.com

October 2000

Purpose: To provide a timestamp format for Grid performance work.

Rationale: Grid performance work will require accuracy across the whole computing Grid. To providing accuracy will require simple and accurate timestamps that can be generated with little impact on cpu, system, or network load.

Attributes of a Timestamp: A timestamp must have the following attributes:

1. Be accurate
2. Be consistent across the Grid
3. Contain accuracy information (sampling error)
4. Contain synchronization information
5. Have little impact on system and network performance

Dialectic: The dialectics brought forth in discussion include:

1. Binary or ASCII timestamps or both? (Brian Tierney)
2. Database or network (UDP TCP) packet format or both? (Dan Gunter)
3. Is the most important item the message or the timestamp? (Dan Gunter)

1. Discussion

Initial discussion among the group members was that an ASCII timestamp would be good for all-around Grid usage. This led me to survey associates on the relative merits of binary and ASCII formats. By and large the binary format was seen as the simplest and the NTP timestamp format offered a well-defined, accurate and accepted timestamp standard.

1.1 ASCII Timestamps

The ASCII timestamp was seen to have some aspects that might make it more difficult to work with than a binary timestamp. No matter what ASCII time stamp was used, it would likely have these attributes:

1. It would need to be converted to a binary format for any sort of performance work, at least internally at the client side. The only place time stamps would be used is inside performance analysis applications and would need to go through a formatting of some sort. If they were going to be reformatted in a print routine or perl script, the presence of a binary timestamp made conversion to an ASCII display format trivial.

2. The presence of any ASCII format would be confusing since it needed to regard:
 1. Conversion to local time from GMT
 2. Daylight savings time
 3. Usual time rollovers, minute, hour, day, month, year, leap year, leap seconds
3. ASCII mostly useful for visual inspection, and even then it can be confusing.

1.2 The Message or the Timestamp?

Which is more important, the message or the timestamp? This issue is one that depends on viewpoint. If one looks at from the programmer's point of view, the timestamp may be seen as a necessary evil. If one looks at from the data integrity side, then accurate timestamps are critical for any performance or archival data.

Dan Gunter pointed out that the "ASCII timestamps are desirable because they can be embedded in a message (in our case a monitored event), not the other way around." He would like timestamps that "can be written to a file, archived in a relational database, stored in a directory service, or transmitted via a higher-level network protocol (like RMI or IIOP). Timestamps should be embeddable any number of times in a given message or data structure. A UDP packet protocol cannot be used for these purposes."

Agreed, that if the data is stored in situ on a server, then yes, there might be no need for a UDP packet. In the case of Grid computing, the impression formed was that was that almost no timestamps would be generated locally [Is this view correct?]. If few timestamps are generated locally, then for performance work there could be many timestamps generated which in turn could quickly overwhelm a system or a network.

Additionally, any archived timestamps may need to be NTP compatible since the synchronization and accuracy need to be recorded with the data. Since the timestamps would be part of a relational database, the need to be human readable may not exist. For a database, the UDP packet could be a clean, compact, and consistent entry. It might be best if the full IP packet were used since it contains the source IP address.

Dan Gunter also wanted to "discuss these issues at a higher level of abstraction, first to stimulate discussion, and second to ease mapping the results of the discussion into more than one syntax." This opens the question of possible multiple formats if there are to be multiple syntaxes, and conversion and correlation between them. As a counterpoint, all data archived with present syntaxes will need to undergo conversion to new syntaxes as they obsolesce older formats. Having multiple formats increases the amount work needed to correlate data, and may not be worthwhile.

The goal in any timestamp format should be to minimize the work for all consumers of time critical data, and maximize reuse.

Another key issue in timestamp error is the construction and issuance of the timestamp by the kernel. Kernel jitter is key and creating a timestamp with ASCII data could substantially increase the timing jitter because of the high overheads in construction of ASCII timestamps from internal clocks. Without the use of NTP compliant clocks, the accuracy of any timestamps become questionable, and at worst unusable for research, especially for Grid computing.

1.3 Network Based Timestamps

Hooking the timestamps to user socket ports was seen as a trivial method to communicate to remote systems. Ask yourself:

Q. How does one transmit a timestamp from one system to another across the Grid?

A. The methods available include:

1. Email -- for simple things like job completions
2. FTP of a file -- transfer of output from a run
3. Telnet or rlogin -- simple remote queries
4. Sockets -- low level communications used by TCP, UDP, which in turn are used by HTTP, XML, RMI, RPC, etc.

XML and RMI are higher-level applications that use sockets to communicate. The only communications that may be reasonable for transmitting application timestamps in the Grid is sockets. [Should other methods be considered?] If one is to use sockets, then a trivial sockets based protocol needs to be proposed.

This leads to a straw man proposal based on RFC 2030 for SNTP (<http://www.landfield.com/rfcs/rfc2030.html>): GTP (Grid Time Protocol) Timestamp Format

2. GTP (Grid Time protocol) Timestamp Format

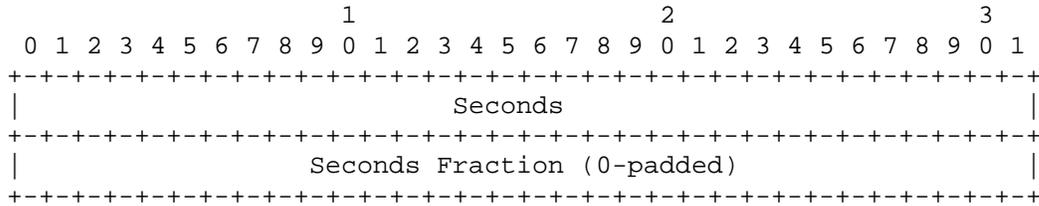
GTP proposes the use of the standard NTP timestamp format described in RFC-1305 and previous versions of that document. In conformance with standard Internet practice, NTP data are specified as integer or fixed-point quantities, with bits numbered in big-endian fashion from 0 starting at the left, or high-order, position. Unless specified otherwise, all quantities are unsigned and may occupy the full field width with an implied 0 preceding bit 0.

Since GTP timestamps are cherished data and, in fact, represent the main product of the protocol, a special timestamp format has been established. GTP timestamps are represented as a 64-bit unsigned fixed-point number, in seconds relative to 0h on 1 January 1900. The integer part is in the first 32 bits and the fraction part in the last 32 bits. In the fraction part, the non-significant low order can be set to 0.

It is advisable to fill the non-significant low order bits of the timestamp with a random, unbiased bitstring, both to avoid systematic

roundoff errors and as a means of loop detection and replay detection (see below). One way of doing this is to generate a random bitstring in a 64-bit word, then perform an arithmetic right shift a number of bits equal to the number of significant bits of the timestamp, then add the result to the original timestamp.

This format allows convenient multiple-precision arithmetic and conversion to UDP/TIME representation (seconds), but does complicate the conversion to ICMP Timestamp message representation, which is in milliseconds. The maximum number that can be represented is 4,294,967,295 seconds with a precision of about 200 picoseconds, which should be adequate for even the most exotic requirements.



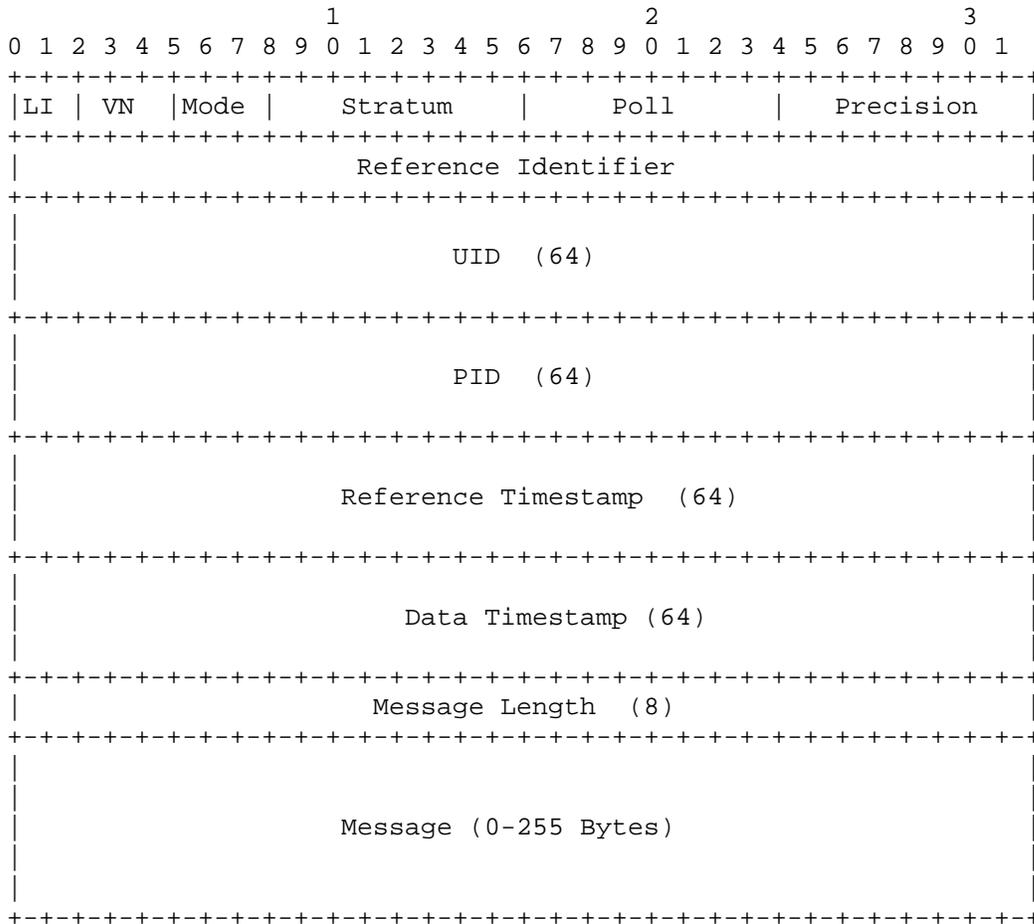
Note that, since some time in 1968 (second 2,147,483,648) the most significant bit (bit 0 of the integer part) has been set and that the 64-bit field will overflow some time in 2036 (second 4,294,967,296). Should GTP or NTP be in use in 2036, some external means will be necessary to qualify time relative to 1900 and time relative to 2036 (and other multiples of 136 years). There will exist a 200-picosecond interval, henceforth ignored, every 136 years when the 64-bit field will be 0, which by convention is interpreted as an invalid or unavailable timestamp.

As the NTP timestamp format has been in use for the last 17 years, it remains a possibility that it will be in use 40 years from now when the seconds field overflows. As it is probably inappropriate to archive GTP timestamps before bit 0 was set in 1968, a convenient way to extend the useful life of GTP timestamps is the following convention: If bit 0 is set, the UTC time is in the range 1968-2036 and UTC time is reckoned from 0h 0m 0s UTC on 1 January 1900. If bit 0 is not set, the time is in the range 2036- 2104 and UTC time is reckoned from 6h 28m 16s UTC on 7 February 2036. Note that when calculating the correspondence, 2000 is not a leap year. Note also that leap seconds are not counted in the reckoning.

2.1 GTP Message Format

Both GTP and NTP are clients of the User Datagram Protocol (UDP) [POS80], which itself is a client of the Internet Protocol (IP) [DAR81]. The structure of the IP and UDP headers is described in the cited specification documents and will not be detailed further here. The UDP port number assigned to GTP is [TBD], which should be used in both the Source Port and Destination Port fields in the UDP header. The remaining UDP header fields should be set as described in the specification.

Below is a description of the proposed GTP message format, which follows the IP and UDP headers. This format is based on that described in RFC-1305. The header fields are defined as follows:



As described in the next section, in Sntp most of these fields are initialized with pre-specified data. For completeness, the function of each field is briefly summarized below.

Leap Indicator (LI): This is a two-bit code warning of an impending leap second to be inserted/deleted in the last minute of the current day, with bit 0 and bit 1, respectively, coded as follows:

LI	Value	Meaning
00	0	no warning
01	1	last minute has 61 seconds
10	2	last minute has 59 seconds)
11	3	alarm condition (clock not synchronized)

Version Number (VN): This is a three-bit integer indicating the GTP version number. The version number is 3 for Version 3 (IPv4 only) and 4 for Version 4 (IPv4, IPv6 and OSI). If necessary to distinguish between IPv4, IPv6 and OSI, the encapsulating context must be inspected.

Mode: This is a three-bit integer indicating the mode, with values defined as follows:

Mode	Meaning
0	reserved
1	symmetric active
2	symmetric passive
3	client
4	server
5	broadcast
6	reserved for GTP control message
7	reserved for private use

In unicast and anycast modes, the client sets this field to 3 (client) in the request and the server sets it to 4 (server) in the reply. In multicast mode, the server sets this field to 5 (broadcast).

Stratum: This is an eight-bit unsigned integer indicating the stratum level of the local clock, with values defined as follows:

Stratum	Meaning
0	unspecified or unavailable
1	primary reference (e.g., radio clock)
2-15	secondary reference (via NTP or SNTP)
16-255	reserved

Poll Interval: This is an eight-bit signed integer indicating the maximum interval between successive messages, in seconds to the nearest power of two. The values that can appear in this field presently range from 4 (16 s) to 14 (16284 s); however, most applications use only the sub-range 6 (64 s) to 10 (1024 s).

Precision: This is an eight-bit signed integer indicating the precision of the local clock, in seconds to the nearest power of two.

The values that normally appear in this field range from -6 for mains-frequency clocks to -20 for microsecond clocks found in some workstations.

Reference Identifier: This is a 32-bit bitstring identifying the particular reference source. In the case of NTP Version 3 or Version 4 stratum-0 (unspecified) or stratum-1 (primary) servers, this is a four-character ASCII string, left justified and zero padded to 32 bits. In NTP Version 3 secondary servers, this is the 32-bit IPv4 address of the reference source. In NTP Version 4 secondary servers, this is the low order 32 bits of the latest transmit timestamp of the reference source. NTP primary (stratum 1) servers should set this field to a code identifying the external reference source according to the following list. If the external reference is one of those listed, the associated code should be used. Codes for sources not listed can be contrived as appropriate.

Code	External Reference Source
LOCL	uncalibrated local clock used as a primary reference for a subnet without external means of synchronization
PPS	atomic clock or other pulse-per-second source individually calibrated to national standards
ACTS	NIST dialup modem service
USNO	USNO modem service
PTB	PTB (Germany) modem service
TDF	Allouis (France) Radio 164 kHz
DCF	Mainflingen (Germany) Radio 77.5 kHz
MSF	Rugby (UK) Radio 60 kHz
WWV	Ft. Collins (US) Radio 2.5, 5, 10, 15, 20 MHz
WWVB	Boulder (US) Radio 60 kHz
WWVH	Kauai Hawaii (US) Radio 2.5, 5, 10, 15 MHz
CHU	Ottawa (Canada) Radio 3330, 7335, 14670 kHz
LORC	LORAN-C radionavigation system
OMEG	OMEGA radionavigation system
GPS	Global Positioning Service
GOES	Geostationary Orbit Environment Satellite

Reference Timestamp: This is the time at which the local clock was last set or corrected, in 64-bit timestamp format.

Server UID: This is a standard UID number in 64-bit format.

Process number: This is a 64-bit process ID used by the kernel to direct the data to the correct process.

Data Timestamp: This is the time at which the client request was satisfied by the server, in 64-bit timestamp format.

Message length: Length of the server side application message in 8-bit format.

Application message: The application message, up to 256 bytes.

2.2 GTP Client Operations

A GTP client can operate in multicast mode, unicast mode or anycast mode. In multicast mode, the client sends no request and waits for a broadcast (mode 5) from a designated multicast server. In unicast mode, the client sends a request (mode 3) to a designated unicast server and expects a reply (mode 4) from that server. In anycast mode, the client sends a request (mode 3) to a designated local broadcast or multicast group address and expects a reply (mode 4) from one or more anycast servers. The client uses the first reply received to establish the particular server for subsequent unicast operations. Later replies from this server (duplicates) or any other server are ignored. Other than the selection of address in the request, the operations of anycast and unicast clients are identical.

Requests are normally sent at intervals from 100ms to 1024 s, depending on the frequency tolerance of the client clock and the required accuracy.

A unicast or anycast client initializes the GTP message header, sends the request to the server and zeros the Data Timestamp field of the reply. For this purpose, all but the UID and PID GTP header fields shown above can be set to 0. In the first octet, the LI field is set to 0 (no warning) and the Mode field is set to 3 (client). The VN field must agree with the version number of the GTP server; however, Version 4 servers will also accept previous versions. Version 3 (RFC-1305) and Version 2 (RFC-1119) servers already accept all previous versions, including Version 1 (RFC-1059).

Since there will probably continue to be GTP servers of all four versions interoperating in the Internet, careful consideration should be given to the version used by GTP Version 4 clients. It is recommended that clients use the latest version known to be supported by the selected server in the interest of the highest accuracy and reliability. SNTTP Version 4 clients can interoperate with all previous version GTP servers, since the header fields used by GTP clients are unchanged. Version 4 servers are required to reply in the same version as the request, so the VN field of the request also specifies the version of the reply.

The following table summarizes the GTP client operations in unicast, anycast and multicast modes. The recommended error checks are shown in the Reply and Multicast columns in the table. The message should be considered valid only if all the fields shown contain values in the respective ranges. Whether to believe the message if one or more of the fields marked "ignore" contain invalid values is at the discretion of the implementation.

Field Name	Request	Unicast/Anycast Reply	Multicast Reply
LI	0	0-2	0-2
VN	1-4	copied from request	1-4
Mode	3	4	5
Stratum	0	1-14	1-14
Poll	0	ignore	ignore
Precision	0	ignore	ignore
Reference Identifier	0	ignore	ignore
Reference Timestamp	0	ignore	ignore
Data Timestamp	0	0	0

2.3 GTP Server Operations

A GTP Version 4 server operating with either a GTP client of the same or previous versions retains no persistent state.

A GTP server can operate in unicast mode, anycast mode, multicast mode or any combination of these modes. In unicast and anycast modes, the server receives a request (mode 3), modifies certain fields in the GTP

header, and sends a reply (mode 4), possibly using the same message buffer as the request. In anycast mode, the server listens on the designated local broadcast or multicast group address assigned by the IANA, but uses its own unicast address in the source address field of the reply. Other than the selection of address in the reply, the operations of anycast and unicast servers are identical. Multicast messages are normally sent at poll intervals from 64 s to 1024 s, depending on the expected frequency tolerance of the client clocks and the required accuracy.

In unicast and anycast modes, the VN and Poll fields of the request are copied intact to the reply. If the Mode field of the request is 3 (client), it is set to 4 (server) in the reply; otherwise, this field is set to 2 (symmetric passive) in order to conform to the GTP specification. This allows clients configured in symmetric active (mode 1) to interoperate successfully, even if configured in possibly suboptimal ways. In multicast (unsolicited) mode, the VN field is set to 4, the Mode field is set to 5 (broadcast), and the Poll field set to the nearest integer base-2 logarithm of the poll interval.

In unicast and anycast modes, the server may or may not respond if not synchronized to a correctly operating radio clock, but the preferred option is to respond, since this allows reachability to be determined regardless of synchronization state. In multicast mode, the server sends broadcasts only if synchronized to a correctly operating reference clock.

The remaining fields of the GTP header are set in the following way. Assuming the server is synchronized to a radio clock or other primary reference source and operating correctly, the LI field is set to 0 and the Stratum field is set to 1 (primary server); if not, the Stratum field is set to 0 and the LI field is set to 3. The Precision field is set to reflect the maximum reading error of the local clock.

For all practical cases it is computed as the negative of the number of significant bits to the right of the decimal point in the GTP timestamp format.

The timestamp fields are set as follows. If the server is unsynchronized or first coming up, all timestamp fields are set to zero. If synchronized, the Reference Timestamp is set to the time the last update was received from the radio clock or modem. In unicast and anycast modes, Data Timestamp fields are set to the time of day when the message is sent. In multicast mode, the Originate Timestamp and Receive Timestamp fields are set to 0 and the Transmit Timestamp field is set to the time of day when the message is sent.

The following table summarizes these actions.

Field Name	3. Request	Unicast/Anycast Reply	Multicast Reply
LI	ignore	0 or 3	0 or 3
VN	1-4	copied from request	4
Mode	3	2 or 4	5
Stratum	ignore	1	1
Poll	ignore	copied from request	log2 poll interval
Precision	ignore	-log2 server significant bits	-log2 server significant bits
Reference Identifier	ignore	source ident	source ident
Reference Timestamp	ignore	time of last radio update copied from request	time of last radio update copied from request
PID/UID	use	copied from request	copied from request
Data Timestamp	ignore	copied from transmit timestamp	0

There is some latitude on the part of most clients to forgive invalid timestamps, such as might occur when first coming up or during periods when the primary reference source is inoperative. The most important indicator of an unhealthy server is the LI field, in which a value of 3 indicates an unsynchronized condition. When this value is displayed, clients should discard the server message, regardless of the contents of other fields.