Simple Network Time Protocol Technical Reference

Interniche Legacy Document

Version 1.00

Date: 18-May-2017 12:03

Table of Contents

1 Introduction

The InterNiche SNTPv4 client implements RFC 4330, providing various APIs that allow an external application task to obtain the time (also referred to as a synchronization request) from a time server.

The SNTPv4 client can request time from unicast-, multicast-, or broadcast-based IPv4 servers. A unicast-based server is one where the client contacts a server at the latter's unicast address. A multicast- or broadcast-based server is defined as one that either periodically transmits the current time to the multicast or broadcast address, or one that responds to client requests addressed to those addresses.

2 Tasks

In a multitasking operating system, the SNTPv4 client code runs in the context of various tasks. The list of tasks that the code runs in includes the SNTPv4 client task, timer task (`tk_nettick`), and the external application task.

The external application task is typically responsible for the following tasks:

1. Providing configuration information (e.g. IPv4 address of server),
2. Requesting the client to obtain time from a server,
3. Terminating a synchronization request, and
4. Obtaining current time based on last update from server and the elapsed time interval since then (based on information extracted from the local clock).

As its name implies, the timer task is responsible for all time-related functionality of the SNTPv4 client. This includes things such as retranmission of a SNTPv4 request if a previous request has not elicited any response. Note that the retransmission could be a repeat request to the "current" server, or it could be the first transmission to a different server or servers.

The SNTPv4 client task processes all of the received response packets and also performs some housekeeping tasks.

3 Mutual Exclusion

Since the SNTPv4 client code executes in the context of multiple tasks, it requires mechanisms to protect the integrity of shared data structures. These data structures include the state machine that directs the operations of the client.

The SNTPv4 client code requires one mutual exclusion primitive (SNTPV4C_RESID) for its proper operation. This is accomplished via LOCK_NET_RESOURCE and UNLOCK_NET_RESOURCE invocations to obtain and release the primitive (e.g., semaphore) as appropriate.

4 Modes of Operation

The SNTPv4 client can be configured to operate in one of various modes.

If the configuration only contains unicast addresses of time servers, the client attempts to contact the servers in the order that they are listed. If it does not receive any response from a server, it starts afresh with the next server in the list.

If the configuration contains a mixture of unicast and non-unicast addresses, the client attempts to contact the unicast servers first (as explained in the previous paragraph). If it does not receive any reply from any of the unicast addresses, the client transitions to using the multicast- and/or broadcast-based servers. However, the mode in which it attempts to obtain time from these servers depends upon the settings of the bm_active and bm_perpetual fields. If the bm_active field is set to a non-zero value, the client transmits one request request to each of the configured multicast or broadcast addresses. (Note that these requests are sent back-to-back.) In this case, the value of the bm_perpetual field is irrelevant. If the bm_active field is set to zero, the client has been configured for passive mode. Passive mode, in turn, comes in two flavors: PASV_ONCE and PASV_PERPETUAL. In the PASV_ONCE mode, the client creates the appropriate set of one or more sockets, and waits for the receipt of a (unsolicited) response from a time server. If no response is received in pasv_tmo amount of time, the sockets are closed, and the synchronization attempt is declared a failure. In the PASV_PERPETUAL mode, the client waits forever for a response from the server. This mode allows for automated operation from the external application's perspective, since it just needs to prime the client with configuration once, and the client automatically picks up the latest time updates from the server.

The receipt of a "kiss-of-death" from a unicast address-based server immediately causes the client code to transition to the next server in the list. The server is also moved to a lower slot in the address table where it will be contacted only if other servers ("ahead" of it in the priority list) do not respond.

5 Data Structures

The SNTPv4 client requires the following data structure to be configured for its operation. The usage of each field is explained below.

```
struct sntpv4_cfg_addr_sp
{
   struct srv_addrs addru;
   u_long  def_delay;
   ip_addr ipv4_bm_accept_srv_addrs [NUM_SRVS_PER_GRP];
#ifdef IP_MULTICAST
   u_char  ttl;
#endif
   u_char sord;
};
```

`addru`	Server identification (via hostname and/or IPv4 address).
`def_delay`	This field can be used to set the one-way delay (in microseconds) to the server. This information is required for multicast- or broadcast-based servers that do not respond to requests addressed to their unicast address. It can also be utilized in scenarios when the dynamic delay cannot be computed (e.g., due to a `SNTPV4_DELTA_TIME_ERR`).
`ipv4_bm_accept_srv_addrs`	This field contains the IPv4 addresses of multicast- or broadcast-based servers from whom we will accept a time update. These addresses must be specified in host byte order.
`ttl`	This field is only relevant for servers that are accessible via multicast addresses. It is used to set the TTL field of the UDP/IP datagram that contains the client's request.
`sord`	This is a three-valued flag. If set to `USE_DYN_DELAY`, the client will attempt to dynamically obtain the round-trip delay (from which it will compute the one-way delay). If the `sntpv4_process_time_info()` function encounters a `SNTPV4_DELTA_TIME_ERR` error when processing a time update (for a `USE_DYN_DELAY` configuration), it returns a failure. If set to `USE_STATIC_DELAY`, the client will use the static value provided in the `def_delay` field. Time computations that use a static delay are always successful. If set to `USE_DYNSTAT_DELAY`), the client will use the dynamically-computed delay value whenever possible, and the static delay when the dynamic delay cannot be computed (e.g., due to a `SNTPV4_DELTA_TIME_ERR`).

```
struct sntpv4_cfg
{
   u_short retx_tmo;
   u_short total_tx;
   u_short pasv_tmo;
   u_char  bm_active;
   u_char  bm_perpetual;
   struct sntpv4_cfg_addr_sp addr_sp [NUM_IPV4_ADDRS];
};
```

`retx_tmo`	Time interval (in units of seconds) between successive transmissions of SNTPv4 requests by the client.
`total_tx`	Maximum number of transmissions of a request to a server or set of multicast- or broadcast-based servers, including the original one.
`pasv_tmo`	Maximum amount of time (in units of seconds) to wait passively for the arrival of a server's response before giving up and declaring the synchronization attempt a failure.
`bm_active`	Flag that controls whether multicast- or broadcast-based servers are queried actively (when set to a value of 1), or waited for passively (when set to a value of 2).
`bm_perpetual`	Flag that controls whether the client listens forever for synchronization updates on broadcast or multicast addresses (when set to a value of 1), or use those addresses for one transaction only (when set to a value of 2). This field is only used when '`bm_active`' indicates passive mode.
`addr_sp`	Server address-specific configuration parameters (e.g., TTL).

```
/* SNTPV4 server address specification data structure */
struct srv_addrs
{
   ip_addr ipv4_addrs;
   char * srv_name;
};
```

`ipv4_addrs`	IPv4 address of time server. The address must be specified in host byte order. The various types of addresses (unicast, multicast, or broadcast) can be present in any order (in the address table).
`srv_name`	DNS hostname of time server. The hostname is resolved via DNS to an IPv4 address, and the latter is used to communicate with the time server.
Note: `Both ipv4_addrs` and `srv_name` fields can be populated with the IPv4 address and hostname of a time server. If only one field is populated, the other must be set to zero.	

6 API

These APIs are intended to be invoked by an external application task that first configures and then uses the SNTPv4 client module to obtain the time from a time server.

6.1 sntpv4_get_time

Name

`sntpv4_get_time`

Syntax

```
int sntpv4_get_time (struct ntp_timestamp * timep);
```

Description

This function is used to obtain the current time from the SNTPv4 client. Note that unlike sntpv4_sync (), this function does not actually initiate a synchronization request. Instead, it computes the current time as the sum of the last update received from the server, and the elapsed time interval (as computed via the local clock (cticks)).

Returns

`SNTPV4_ERR`	if the client has never obtained a time from a time server
`SNTPV4_OK`	if the current time info has been computed and copied into `*timep`

6.2 sntpv4_get_time_unix

Name

`sntpv4_get_time_unix`

Syntax

```
int sntpv4_get_time_unix (struct timeval_u * tvp);
```

Description

This function is identical in its functionality to sntpv4_get_time(), with the only difference being that it returns the current time in a Unix-like time structure (instead of the NTP timestamp format). A value of `0` corresponds to 0:00 hours on January 1, 1970.

Returns

`SNTPV4_ERR`	if the client has never obtained a time from a time server
`SNTPV4_OK`	if the current time info has been computed and copied into `*tvp`

6.3 sntpv4_init

Name

`sntpv4_init`

Syntax

```
int sntpv4_init (void);
```

Description

This function is intended to be called during the TCP/IP stack's initialization. It initializes the SNTPv4 client state machine.

Returns

`SNTPV4_ERR`	on error
`SNTPV4_OK`	on success

6.4 sntpv4_retrieve_all

Name

`sntpv4_retrieve_all`

Syntax

```
int sntpv4_retrieve_all (struct sntpv4_cfg * cfgp, struct sntpv4_state *
statep, struct sntpv4_last_update * updp);
```

Description

This function retrieves the current configuration, state, and last update (from the server) from the SNTP client's internal data structures. Data will not be copied for any information whose parameter value is NULL.

Returns

`SNTPV4_OK`	after copying all internal data into application structures.

6.5 sntpv4_set_cfg

Name

`sntpv4_set_cfg`

Syntax

```
int sntpv4_set_cfg (struct sntpv4_cfg * cfgp);
```

Description

This function is used to provide configuration information to the SNTPv4 client.

Returns

`SNTPV4_BUSY_ERR`	client is busy and unable to process
`SNTPV4_BAD_CFG_ERR`	illegal or invalid configuration.
`SNTPV4_OK`	on success

6.6 sntpv4_sync

Name

`sntpv4_sync`

Syntax

```
int sntpv4_sync (void);
```

Description

This function is used to request the client to initiate a synchronization attempt with the time server.

Returns

`SNTPV4_OK`	success
`SNTPV4_SYNC_IN_PROG_ERR`	sync request is already in progress
`value < 0`	See file `sntpv4.h` for specific error detail

6.7 sntpv4_term

Name

`sntpv4_term`

Syntax

```
int sntpv4_term (void);
```

Description

This function is used to terminate any ongoing synchronization session.

Returns

`SNTPV4_NO_CFG_ERR`	if the client has not been configured
`SNTPV4_SYNC_NOT_IN_PROG_ERR`	if the client was idle
`SNTPV4_OK`	on success

7 Logging

The SNTPv4 client code contains support for a logging mechanism that captures all state transitions inside the client. Each state transition event is logged with the time when it occurred, and an identifier for the C language function where it occurred. The logging API also allows for an additional three 32-bit quantities to be logged along with each entry.

The client code also contains support for conditional debug printfs that provide information on the execution of the code.

8 CLI Commands

The SNTPv4 client code provides support for the following CLI commands:

8.1 sntp netstat

Command Name

`sntp netstat - display SNTPv4 client statistics and status`

Syntax

`netstat`

Parameters

None

Description

This command displays statistics associated with the SNTPv4 client module.

Location

This command is provided by the `SNTPv4 client` module when `USE_SNTP_V4` is defined.

8.2 sntp sntpcfg

Command Name

`sntp sntpcfg - configure SNTPv4 client`

Syntax

`sntpcfg -a -c -d -e -l -m -o -p -r -s -t -v -x`

Parameters

`-a`	Add new time server address (specified as IPv4 address or hostname).
`-c`	Specify list of time servers from which an update will be accepted.
`-d`	Delete time server address (specified as IPv4 address or hostname).
`-e`	Specify value of the "bm_perp" parameter ("none", "once", or "perpetual") for time server.
`-l`	Specify value of "def_delay" parameter (in microseconds) for time server.
`-m`	Specify value of the "bm_active" parameter ("none", "active", or "passive") for time server.
`-o`	Specify value of the "sord" parameter ("none", "dynamic", or "static") for time server.
`-p`	Specify slot number in time server address table.
`-r`	Specify retransmission timeout (in seconds).
`-s`	Specify time server (as IPv4 address or hostname) for which parameters are being configured.
`-t`	Specify value of TTL field in IPv4 header for outgoing multicast packets.
`-v`	Specify passive timeout (in seconds).
`-x`	Specify total number of transmissions of request packet.

Description

This command is used to configure the SNTPv4 client module.

Notes/Status

- The -p option specifies the slot number for the time server being added via the `-a` option. Both of these options must be specified together.
- The -c, -l, -o, and -t options must be used to configure parameters for an existing time server specified via the `-s` option.
- The -c, -e, -l, -m, and -o options are only intended for use with broadcast- or multicast-based time servers.

The following sequence of CLI commands can be used to configure the SNTPv4 client to communicate with a unicast-based time server.

##add a new time server (129.6.15.28) at slot# zero in the server address table
```
sntpcfg -a 129.6.15.28 -p 0
```

Alternatively, a server can be specified via its name.

```
sntpcfg -a time-a.nist.gov -p 0
```

The following sequence of CLI commands can be used to configure the SNTPv4 client to operate in passive perpetual mode, and to accept updates sent from 10.0.0.70 to the 224.0.1.1 multicast group address. The server is configured for a static delay of 10000 microseconds.

##add a new time server (224.0.1.1) at slot# zero in the server address table
```
sntpcfg -a 224.0.1.1 -p 0
```

##configure client to use a static one-way delay of 10000 microseconds
```
sntpcfg -s 224.0.1.1 -o static -l 10000
```

##configure client to operate in passive perpetual mode
```
sntpcfg -m passive -e perpetual
```

##configure client to only accept updates sent from 10.0.0.70 to the 224.0.1.1 multicast group address
```
sntpcfg -s 224.0.1.1 -c 10.0.0.70
```

Broadcast-based time servers can be configured in a manner similar to that shown above. To configure a client for operation in active mode (with multicast- or broadcast-based servers), use the `'-m active'` option. Here's an example of configuration that allows the client to accept time updates (sent to the 10.0.0.255 broadcast address) from any one of four servers.

```
sntpcfg -s 10.0.0.255 -c "10.0.0.1 10.0.0.2 10.0.0.3 10.0.0.4"
```

Location

This command is provided by the `SNTPv4 client` module when `USE_SNTP_V4` is defined.

8.3 sntp sntplog

Command Name

`sntp sntplog - display SNTPv4 client state transition log`

Syntax

`sntplog`

Parameters

None

Description

This command displays the contents of the state transition log in the time server.

Location

This command is provided by the `SNTPv4 client` module when `USE_SNTP_V4` is defined.

8.4 sntp sntpsync

Command Name

`sntp sntpsync - synchronize with the time server`

Syntax

`sntpsync`

Parameters

None

Description

This command initiates a sync with the time server.

Location

This command is provided by the `SNTPv4 client` module when `USE_SNTP_V4` is defined.

8.5 sntp sntpterm

Command Name

`sntp sntpterm - terminate synchronization with the time server`

Syntax

`sntpterm`

Parameters

None

Description

This command terminates an ongoing sync with the time server.

Location

This command is provided by the `SNTPv4 client` module when `USE_SNTP_V4` is defined.

8.6 sntp sntptime

Command Name

`sntp sntptime - display current time`

Syntax

`sntptime`

Parameters

None

Description

This command displays the current time. The time returned is the sum of the last update received from the time server, and the elapsed interval since then (based on information extracted via the local clock).

Location

This command is provided by the `SNTPv4 client` module when `USE_SNTP_V4` is defined.

9 Feature Control

Compile time control of the SNTPv4 client can be broken in to three different categories: those required for inclusion and operation; those controlling limits and sizes; and those enabling debugging or other optional features.

The following `#defines` are required for operation of the SNTPv4 client module.

```
#define USE_SNTP_V4             1  /* SNTPv4 client module (RFC 4330) */
#define UDP_SKIP_LCL_ADDR_CHECK 1  /* skip checking for local address validity when
                                      performing bind () on datagram socket */
#define MAC_LOOPBACK            1  /* user MAC layer pseudo-Ethernet loopback driver */
```

These `#defines` control limits, sizes and threshholds:

```
#define NUM_IPV4_ADDRS          4  /* number of IPv4 time servers to be acted upon */
#define NUM_SRVS_PER_GRP        4  /* max multicast/broadcast servers */
```

These `#defines` affect the operation of SNTP and may be appropriate for your particular application. They are most likely found in the `ipport.h` file.

```
#define IP_MULTICAST            1  /* support IP multicast capability in the stack */
#define DNS_CLIENT              1  /* include DNS client code */
```

If it is necessary to debug operation of the SNTPv4 client, these `#defines` may be useful.

```
#define SNTPV4_DEBUG            1  /* SNTPv4 client module - enable debug printfs */
#define SNTPV4_LOGGING          1  /* SNTPv4 client module - enable logging of state
transitions */
#define SNTPV4_COUNTERS         1  /* SNTPv4 client module - enable collection of statistics */
```

10 Source Code Files

The SNTPv4 client source and header files are located in `sntp/sntpv4.c` and `sntp/sntpv4.h` respectively.

11 Sample Application

The SNTPv4 external application that drives and configures the SNTPv4 client module is located in `sntp/sntpv4_app.c`.