NAME

pcap-tstamp − packet time stamps in libpcap

DESCRIPTION

When capturing traffic, each packet is given a time stamp representing, for incoming packets, the arrival time of the packet and, for outgoing packets, the transmission time of the packet. This time is an approximation of the arrival or transmission time. If it is supplied by the operating system running on the host on which the capture is being done, there are several reasons why it might not precisely represent the arrival or transmission time:

if the time stamp is applied to the packet when the networking stack receives the packet, the networking stack might not see the packet until an interrupt is delivered for the packet or a timer event causes the networking device driver to poll for packets, and the time stamp might not be applied until the packet has had some processing done by other code in the networking stack, so there might be a significant delay between the time when the last bit of the packet is received by the capture device and when the networking stack time-stamps the packet;

the timer used to generate the time stamps might have low resolution, for example, it might be a timer updated once per host operating system timer tick, with the host operating system timer ticking once every few milliseconds;

a high-resolution timer might use a counter that runs at a rate dependent on the processor clock speed, and that clock speed might be adjusted upwards or downwards over time and the timer might not be able to compensate for all those adjustments;

the host operating system's clock might be adjusted over time to match a time standard to which the host is being synchronized, which might be done by temporarily slowing down or speeding up the clock or by making a single adjustment;

different CPU cores on a multi-core or multi-processor system might be running at different speeds, or might not have time counters all synchronized, so packets time-stamped by different cores might not have consistent time stamps;

some time sources, such as those that supply POSIX "seconds since the Epoch" time, do not count leap seconds, meaning that the seconds portion (**tv_sec**) of the time stamp might not be incremented for a leap second, so that the fraction-of-a-second part of the time stamp might roll over past zero but the second part would not change, or the clock might run slightly more slowly for a period before the leap second.

For these reasons, time differences between packet time stamps will not necessarily accurately reflect the time differences between the receipt or transmission times of the packets.

In addition, packets time-stamped by different cores might be time-stamped in one order and added to the queue of packets for libpcap to read in another order, so time stamps might not be monotonically increasing.

Some capture devices on some platforms can provide time stamps for packets; those time stamps are usually high-resolution time stamps, and are usually applied to the packet when the first or last bit of the packet arrives, and are thus more accurate than time stamps provided by the host operating system. Those time stamps might not, however, be synchronized with the host operating system's clock, so that, for example, the time stamp of a packet might not correspond to the time stamp of an event on the host triggered by the arrival of that packet. If they are synchronized with the host operating system's clock, some of the issues listed above with time stamps supplied by the host operating system may also apply to time stamps supplied by the capture device.

Depending on the capture device and the software on the host, libpcap might allow different types of time stamp to be used. The **pcap_list_tstamp_types**(3PCAP) routine provides, for a packet capture handle created by **pcap_create**(3PCAP) but not yet activated by **pcap_activate**(3PCAP), a list of time stamp types supported by the capture device for that handle. The list might be empty, in which case no choice of time stamp type is offered for that capture device. If the list is not empty, the **pcap_set_tstamp_type**(3PCAP) routine can be used after a **pcap_create**() call and before a **pcap_activate**() call to specify the type of time

stamp to be used on the device. The time stamp types are listed here; the first value is the #define to use in code, the second value is the value returned by **pcap_tstamp_type_val_to_name**(3PCAP) and accepted by **pcap_tstamp_type_name_to_val**(3PCAP)**.**

PCAP_TSTAMP_HOST - **host**
Time stamp provided by the host on which the capture is being done. The precision of this time stamp is unspecified; it might or might not be synchronized with the host operating system's clock.

PCAP_TSTAMP_HOST_LOWPREC - **host_lowprec**
Time stamp provided by the host on which the capture is being done. This is a low-precision time stamp, synchronized with the host operating system's clock.

PCAP_TSTAMP_HOST_HIPREC - **host_hiprec**
Time stamp provided by the host on which the capture is being done. This is a high-precision time stamp, synchronized with the host operating system's clock. It might be more expensive to fetch than **PCAP_TSTAMP_HOST_LOWPREC**.

PCAP_TSTAMP_HOST_HIPREC_UNSYNCED - **host_hiprec_unsynced**
Time stamp provided by the host on which the capture is being done. This is a high-precision time stamp, not synchronized with the host operating system's clock. It might be more expensive to fetch than **PCAP_TSTAMP_HOST_LOWPREC**.

PCAP_TSTAMP_ADAPTER - **adapter**
Time stamp provided by the network adapter on which the capture is being done. This is a high-precision time stamp, synchronized with the host operating system's clock.

PCAP_TSTAMP_ADAPTER_UNSYNCED - **adapter_unsynced**
Time stamp provided by the network adapter on which the capture is being done. This is a high-precision time stamp; it is not synchronized with the host operating system's clock.

Time stamps synchronized with the system clock can go backwards, as the system clock can go backwards. If a clock is not in sync with the system clock, that could be because the system clock isn't keeping accurate time, because the other clock isn't keeping accurate time, or both.

Host-provided time stamps generally correspond to the time when the time-stamping code sees the packet; this could be some unknown amount of time after the first or last bit of the packet is received by the network adapter, due to batching of interrupts for packet arrival, queueing delays, etc..

By default, when performing a live capture or reading from a savefile, time stamps are supplied as seconds since January 1, 1970, 00:00:00 UTC, and microseconds since that seconds value, even if higher-resolution time stamps are available from the capture device or in the savefile. If, when reading a savefile, the time stamps in the file have a higher resolution than one microsecond, the additional digits of resolution are discarded.

The **pcap_set_tstamp_precision**(3PCAP) routine can be used after a **pcap_create**() call and after a **pcap_activate**() call to specify the resolution of the time stamps to get for the device. If the hardware or software cannot supply a higher-resolution time stamp, the **pcap_set_tstamp_precision**() call will fail, and the time stamps supplied after the **pcap_activate**() call will have microsecond resolution.

When opening a savefile, the **pcap_open_offline_with_tstamp_precision**(3PCAP) and **pcap_fopen_offline_with_tstamp_precision**(3PCAP) routines can be used to specify the resolution of time stamps to be read from the file; if the time stamps in the file have a lower resolution, the fraction-of-a-second portion of the time stamps will be scaled to the specified resolution.

The **pcap_get_tstamp_precision**(3PCAP) routine returns the resolution of time stamps that will be supplied; when capturing packets, this does not reflect the actual precision of the time stamp supplied by the hardware or operating system and, when reading a savefile, this does not indicate the actual precision of time stamps in the file.

SEE ALSO

pcap(3PCAP)