Offline Time Synchronization for libpcap Logs

Daniel Marks, Wolfgang Kiess, Björn Scheuermann, Magnus Roos, Martin Mauve, and Florian Jarre

Heinrich Heine University, Düsseldorf, Germany

I. Introduction

A fundamental problem in real-world computer network experiments is that each system uses its own local clock to timestamp events. These clocks are not perfectly accurate, and thus deviate from each other. Event timestamps assigned by different nodes can therefore not immediately be compared, making the analysis of experimental results difficult. The synchronization of the clocks *online* during the experiment is at most a partial solution to the problem. While using high-precision, special-purpose clocks implies high effort and expensive hardware, online time synchronization protocols like NTP [3] require a permanent, reliable network connection between each node and a reference clock. This cannot always be guaranteed during an experiment. Using such a time synchronization protocol also generates network traffic, which might interfere with the traffic of the experiment itself, and therefore potentially influences the results. Furthermore, even if the clocks were perfectly synchronized, it takes some system dependent (and potentially non-deterministic) time from the occurrence of an event until it is actually timestamped and recorded. While it may be possible to use customized hard- and software to bound this delay, such a solution cannot be employed for the off-the-shelf systems often used in network experiments.

In order to avoid these problems we have proposed in earlier work [4] to record the occurring events with the deviating, local clocks and synchronize the resulting event log files offline after the experiment. The synchronization is based on so-called *anchor points*, that is, on events that have been recorded and timestamped by more than one node in parallel. The anchor points allow to set the clocks of the nodes into relation. In networks where the medium has a broadcast characteristic (like many wireless networks, but also Ethernet using hubs), the (almost) parallel reception of a packet transmission by multiple nodes can serve as such an anchor point.

In [4], we laid the foundations of this technique. Here, we go one step further and discuss aspects that arise if it is to be applied in a real network. We introduce `pcapsync`, a tool using the algorithm from [4] to synchronize event logs from experiments in IEEE 802.11 wireless networks. It reads a set of log files that have been recorded in libpcap format (used, e. g., by tcpdump [5] and Wireshark [6]), identifies potential anchor points in them, applies our offline time synchronization algorithm, maps the recorded local timestamps to a common, global time scale, and finally writes back a corresponding set of synchronized libpcap files. Its output can thus immediately be used for further analysis with standard tools.

II. MLE Time Synchronization

The synchronization algorithm used by `pcapsync` has been introduced in [4], here we provide a rough overview. As its input, the algorithm is given a set of events that have each been observed by two or more nodes (i. e., the anchor points), and their local timestamps. These events provide information about multiple nodes' clocks at a common point in time. The output includes estimates for the clock rates and offsets, and a synchronized timestamp for each of the anchor points.

The approach assumes clocks to be linear, which is a good approximation at least for experiment durations of up to about 20 minutes. Clocks are thus characterized by a *rate* r and an *offset* o. When read at "true" time[1] T, the clock shows

$$C(T) = r \cdot T + o. \tag{1}$$

Based on additional assumptions on the timestamping process, a maximum likelihood estimator (MLE) for rates, offsets, and event times can be established. This reduces the synchronization problem to an optimization problem. By substitutions and transformations, the MLE can be expressed as a linear program (LP). However, with an increasing number of nodes and anchor points the matrix of coefficients of the LP soon becomes very big, so that standard LP solvers cannot be applied in a straightforward way.

The matrix is very sparse, though. It can be arranged in a special way such that its structure can be exploited to reduce both computational and storage complexity. We use an interior point method, a variant of Mehrotra's predictor-corrector algorithm [2], to solve the linear program. Analytical, simulative, and experimental evaluations presented in [4] show that the estimate is good even for a relatively limited number of available anchor points, and quickly improves further as their number increases.

III. Pcapsync

To apply MLE timestamp synchronization to real-world experiments in IEEE 802.11 networks, we have implemented the `pcapsync` tool. Figure 1 gives an overview of `pcapsync`'s general operation. In its initial step, it parses sets of libpcap log files and identifies anchor points. Anchor points are the foundation of MLE synchronization, and its performance crucially depends on correctly identifying them. In `pcapsync`, we use parallel receptions of the same transmission as anchor points. For a real wireless network, it is thus necessary to identify groups of timestamped packet receptions in the libpcap files,

[1] An absolute time scale does of course not exist; it is, however, assumed here for simplicity.

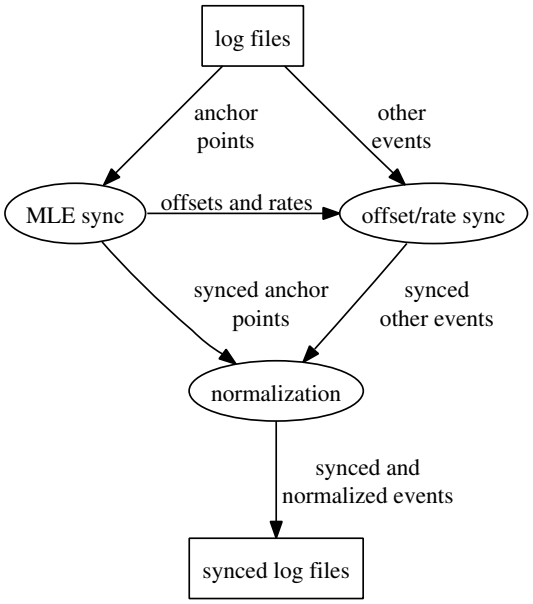

Fig. 1. Structure of pcapsync.

from which it is known for sure that they belong to the same physical transmission. One central duty of `pcapsync` is thus to identify such events.

The link layer reliability mechanism in IEEE 802.11 retransmits unicast packets up to seven times if an acknowledgment is missing [1]. If multiple nodes receive the same unicast transmission[2], these events therefore do not necessarily belong to the same physical layer transmission: for example, as shown in Figure 2, it may well happen that one node receives only the first transmission attempt, while another node receives only the second one. Unfortunately, it is not possible to record the number of performed retransmissions in a hardware-independent way. A packet reception log entry also does not reveal which (re)transmission attempt has been received. Thus, in the specific case of 802.11, multiple receptions of the same unicast packet cannot be used as anchor points.

For broadcast packets there is no automatic retransmission. Still, however, it can happen that identical broadcast packets recorded in the log files refer to different transmissions since higher layers may generate multiple copies of the same packet. ARP, for instance, often broadcasts identical requests when the same address is resolved again. However, broadcast packets generated multiple times can easily be identified using the records about sent packets in the log files. Consequently, they are not used as anchor points. In summary, `pcapsync` is able to use parallel receptions of globally unique broadcast transmissions as anchor points for the synchronization in 802.11 networks.

Based on these rules, the events that can be used as anchor points and those which are not suitable for this purpose can be identified and separated. For the anchor points, synchronized

[2]Note that this is generally possible if the log files are recorded in promiscuous mode.

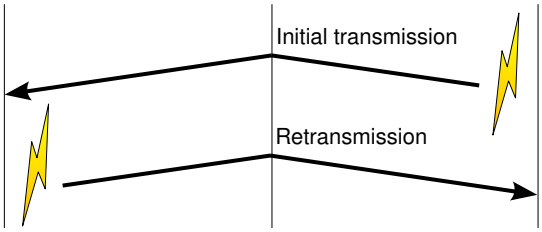

Fig. 2. Ambiguous reception times in case of retransmissions.

timestamps are estimated by the MLE algorithm. As it also yields estimates for clock rates and offsets of all nodes, the timestamps of all other events can be corrected by applying a linear transformation. For an event observed at local time t by some node with estimated clock rate $\widehat{r}$ and estimated offset $\widehat{o}$, it can easily be seen from (1) that the corrected, global timestamp $\widehat{T}$ is given by

$$\widehat{T} = \frac{t - \widehat{o}}{\widehat{r}}. \tag{2}$$

After calculating global timestamps for all events, `pcapsync` normalizes them such that the first event in the experiment occurs at time zero. For this normalization, the globally earliest synchronized event timestamp is subtracted from all timestamps.

Finally, `pcapsync` writes the data to new, synchronized per-node log files. To simplify the evaluation and visualization of an network experiment, it also offers the option to write the synchronized data into one global log file.

IV. Conclusion

In this paper, we have introduced `pcapsync`, a tool for the offline time synchronization of libpcap log files recorded in experiments with IEEE 802.11 networks. This tool is based on MLE timestamp synchronization [4], which uses parallel event observations to relate the clocks of different nodes. We have discussed how suitable events can be identified in real world log data. We have also shown how globally synchronized timestamps can be obtained also for other events. In summary, we believe that `pcapsync` will prove a valuable tool for evaluating experimental results.

Acknowledgments

Part of the work presented here has been supported by the German Research Foundation (DFG).

References

[1] IEEE LAN MAN Standards Committee. ANSI/IEEE Std 802.11, 1999 Edition (R2003), Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
[2] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.
[3] D. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard), Mar. 1992.
[4] B. Scheuermann, W. Kiess, M. Roos, F. Jarre, and M. Mauve. On the time synchronization of distributed log files in networks with local broadcast media. *IEEE/ACM Transactions on Networking*. In press.
[5] tcpdump: a tool for network monitoring, protocol debugging and data acquisition. http://www.tcpdump.org.
[6] The wireshark network protocol analyzer. http://www.wireshark.org/.