Who Said That? – The Send-Receive Correlation Problem in Network Log Analysis

Björn Scheuermann Wolfgang Kiess
Computer Networks Research Group, Heinrich Heine University, Düsseldorf, Germany
{scheuermann, kiess}@cs.uni-duesseldorf.de

1. INTRODUCTION

In computer network experiments, each participating node typically records a local log file using tools like tcpdump. Such files contain the transmitted and received packets along with time stamps. When the outcome of an experiment is analyzed, an interesting problem may arise: given a log entry for a reception, what is the corresponding transmission?

If each transmission is *unique*, in the sense that during the whole recorded time span only one binary identical packet transmission has taken place, this question is trivial. However, with many protocols, binary identical packet transmissions may occur. Typical causes are retransmissions of non-acknowledged data, repeated ARP requests [5], or periodic hello messages (like, e. g., in [2]). Then, the question becomes interesting: from the reception log entry alone, we cannot immediately tell which one of the binary identical transmissions in the log files caused it.

A simple solution is the addition of unique identifiers to each packet, to avoid ambiguities altogether. However, in network experiments, one usually intends to assess the performance of a protocol or application *as-is*. Modifications alter the subject under test and should thus be avoided. Existing literature on events and event orders in distributed systems—beginning with Lamport's milestone paper [3]—can provide starting points. But work in that area typically assumes unique message IDs or even more piggybacked data (like logical clock ticks), raising similar concerns as above. Another option are heuristic solutions based on the time stamps in the logs, along the lines of [1, 4]. There, binary identical packet receptions that fall within a short time interval are simply *assumed* to stem from the same transmission—this may be correct, but it may also be wrong. Many factors may influence the time stamping process [6], so it is hard, if not impossible, to provide bounds on its accuracy (on affordable standard hardware at least). Therefore, any time stamp based approach is *necessarily* a heuristic: it may yield correct solutions, but we can never be completely sure. Here, we ask the more fundamental question: what can we tell *for sure* about the send-receive correlation?

2. LOCAL BROADCAST NETWORKS

We consider this question in networks with local broadcast media, like, for instance, WiFi networks or MANETs. We will assume that propagation delays on the medium are negligible in relation to the duration of a packet transmission; this is the case in practically relevant local broadcast networks. We furthermore assume that each node can be involved in only one successful transmission at a time, and that logging preserves the order of events at the node's interface, i. e., the order of packet transmissions and receptions at the transceiver and in the log file are identical. Such a network exhibits the following properties: (1) a transmission will potentially be recorded by multiple receivers.[1] (2) when two or more nodes record log entries for a number of transmissions (as their sender or as a receiver), then they are logged in the same order.

Essentially, property (2) means that transmissions cannot "overtake" each other on the medium: they may not appear in one order at one node and in a different order at another node. This implies that each local log file provides us with partial information on the overall (i. e., global) order of transmissions. We can use this to "check" a hypothesis about which packet reception stems from which transmission: if we substitute all receptions with their assumed transmissions, we obtain a number of sequences of packet transmission events, one per log file. If our hypothesis is compatible with what has been observed, these sequences must not be contradictory with respect to the event order.

3. AN INTUITIVE EXAMPLE

Let us first illustrate the problem with Figure 1(a) that shows log files from an experiment with three nodes N_1, N_2, N_3: each line corresponds to the local log file of one node (N_1, N_2, N_3), as indicated in the leftmost column. Within each line, events have been recorded in the order from left to right. Capital letters (A, B) denote packet transmissions, corresponding lower case letters (a, b) indicate a matching packet reception. The numerical indices serve the only purpose to allow us to refer to the individual entries.

In the example, all transmissions are initiated by N_1. It first transmits two binary identical packets of type A, then a different packet B, and finally a fourth one that is again binary identical to the first two. Some of these transmissions are received by N_2 and N_3: N_2 first receives a packet that looks like B, then A. N_3 first receives A, then B.

[1] For unicast transmissions, this can be achieved if the interfaces are running in so-called promiscuous mode.

(a) Log files. (b) Intuitive solution.

Figure 1: An example experiment outcome.

Based on the properties of local broadcast networks, human intuition easily solves the puzzle, see Figure 1(b). Since B has been transmitted only once in the entire experiment, it is trivial to see that both b_5 and b_8 are receptions of B_3. Because the order of packets is preserved on the medium, a_6 must stem from a transmission that has been on the medium *after* B_3 (because b_5 and a_6 have occurred in that order, according to N_2's log). Consequently, a_6 belongs to A_4. From N_3's log, we know that a_7 has occurred *before* b_8, and thus before B_3. Therefore, we can exclude that a_7 is a reception of A_4—but we cannot decide between A_1 or A_2.

In summary, taking into account all local log files allows to use logical reasoning to derive the send events where each receive event came from. But we may not expect to be able to always narrow down the set of possibilities to one single transmission—as we can see in the case of a_7 above. Instead, for each receive event in all log files, we aim to determine the *set* of send events that could possibly have been its source. We call this problem the *send-receive correlation problem*.

4. FORMAL PROBLEM STATEMENT

The send-receive correlation problem can be stated as follows. Given are a set of nodes $\mathcal{N}$, a set of packet types $\mathcal{P}$, and a set of log entries L. Each log entry is recorded by a node and refers to one packet type. We capture these relationships through two mappings $N : L \to \mathcal{N}$ and $P : L \to \mathcal{P}$. Furthermore, each log entry is either a send or a receive entry; this is expressed by the mapping $D : L \to \{s, r\}$, which maps each log entry either to s ("send") or to r ("receive").

For convenience, we also introduce the following additional notations. For each $n \in \mathcal{N}$, let L_n be the set of log entries recorded by n, i. e., $L_n := \{l \in L \mid N(l) = n\}$. Let $R := \{l \in L \mid D(l) = r\}$ be the set of all receive events and $S := \{l \in L \mid D(l) = s\}$ the set of all send events.

Recall that the log file of each node is an *ordered* set of event records. Thus, for each node $n \in \mathcal{N}$, we know an irreflexive total order $<_n \subset L_n \times L_n$. For two log entries $l_1, l_2 \in L_n$, it holds that $l_1 <_n l_2$ if and only if l_1 describes an event that happened before l_2. Note that $<_n$ is not a global order of all events, but only describes the partial knowledge about the event order that is given by n's log file.

In summary, the input of the problem consists of $\mathcal{N}$, $\mathcal{P}$, L, N, P, and D, along with the local orders $<_n$ for all $n \in \mathcal{N}$.

For each receive event $x \in R$, our aim is to determine the set S_x of send events that are possible candidates where the received packet could have come from. We call this set the *send candidate set* of x. Defining the exact conditions which a send event $s \in S$ must fulfill in order to be in the set S_x of some receive event $x \in R$ turns out to be tricky. In order to do so, we first introduce the notions of a *global transmission order (GTO)* and a *global assignment (GA)*. A GTO is an irreflexive partial[2] order $\prec$ on the set of all transmission events S (that is, $\prec \subset S \times S$). It can be understood as a hypothesis about the global order of all packet transmissions in the network. Along similar lines, a GA is a hypothesis about the sources of each reception event. It would be conceivable to model a GA as a mapping $R \to S$; for later notational convenience, however, we use a slightly modified variant: a GA is a mapping $L \to S$, where each element in S is mapped to itself (recall that $L = R \cup S$). Furthermore, we require that for all reception events, the assigned send event must be of the same packet type. Finally, a node may not receive its own transmissions. These conditions exclude obviously invalid assignments. In summary, $A : L \to S$ is a GA if and only if (iff) the following conditions hold:

1. $\forall l \in S : A(l) = l$,
2. $\forall l \in L : P(A(l)) = P(l)$, and
3. $\forall l \in R : N(A(l)) \neq N(l)$.

So far, the definitions of a GTO and a GA allow for virtually arbitrary hypotheses. These need not necessarily fit together with the event orders in the log files. We therefore introduce the notion of consistency for a pair $(\prec, A)$ of a GTO and a GA. We call such a pair *consistent* iff

$$\forall n \in \mathcal{N} : \forall l_1, l_2 \in L_n : (l_1 <_n l_2 \Rightarrow A(l_1) \prec A(l_2)).$$

Recall our remark on replacing receive events by their corresponding send events in Sec. 2. This in mind, the above condition becomes clear: if we replace each log entry by the corresponding (according to A) transmission, the assumed global order of transmissions $\prec$ must not contradict any of the local logs. The substitution essentially translates (local) log entry orders to the (global) order of packet transmissions.

Basically, if $(\prec, A)$ is consistent, then it is *possible* that it describes the actual course of events in the experiment. There may be many consistent pairs $(\prec, A)$ for a given set of log files; in that case, we cannot be sure which one is the "truth". All of them are equally possible, because we have only limited information about the true series of events—we only know the recorded events and their local orders.

We can now define consistency for a GA A: A is consistent iff there exists a GTO $\prec$ such that $(\prec, A)$ is consistent. Based on this notion, we can formally capture the sets of send events that, according to the information present in the local log files, could possibly be the source of a given receive event: for $x \in R$, a send event $s \in S$ is in the send candidate set of x iff there exists a consistent GA A such that $A(x) = s$. In that case, based on the log files, we cannot exclude the possibility that s is the transmission that belongs to x—this is exactly what we are looking for.

5. APPLICATIONS

Which transmissions could have been the source of a given packet reception event is a very fundamental question. Thus, the send-receive correlation problem in general has many applications in the analysis of experimental logs. One specific application area is anchor-point based offline time synchronization as discussed in [6] (in fact, this application led us to the problem in the first place). In [6], we consider a set of experiment log files that have been recorded with deviating local clocks. We show how so-called anchor points can be used to eliminate the clock deviations post facto. Anchor points are groups of reception events that stem from the same packet transmission. Our technique is able to provide very accurate time synchronization without any network overhead—if enough anchor points can be identified;

[2]A GTO is a *partial* order because transmissions may take place in parallel if disjoint groups of nodes are involved, for instance in different collision domains of wireless networks.

more anchor points result in better synchronization. Identifying anchor points, however, essentially means solving the sender-receiver correlation problem.

Besides such direct applications, the relation of transmission and reception events is tightly intertwined with the question of the global event order (here expressed by the concept of GTOs): the consistent (GTO,GA)-pairs also show in which order transmissions could have occurred—and which orders can definitely be excluded, because there is no matching (GTO,GA)-pair. Hence, the scope of the problem actually goes significantly beyond the initial question.

6. CALCULATING CANDIDATE SETS

For a finite set of finite log files, there are only a finite number of GTOs and GAs. To determine S_x for a receive event x, we could, in theory, generate all (GTO,GA)-pairs and check them for consistency. Whenever some consistent GA A says that $A(x) = y$, we include y in S_x. However, because the number of possible GTOs and GAs grows exponentially with the problem size, this is prohibitively expensive. Coming up with a better algorithm turns out to be surprisingly difficult; we even suspect NP-hardness.

We do, however, have a solution approach for a slightly simplified problem, with the additional assumption that different nodes may not transmit binary identical packets, i. e.

$$\forall y_1, y_2 \in S : (P(y_1) = P(y_2) \Rightarrow N(y_1) = N(y_2)).$$

This holds, e. g., if each packet contains the sender's address. Then, for each packet type $p \in \mathcal{P}$, there is a unique source node, which we denote by n_p. Consequently, n_p's local order $<_{n_p}$ induces a total order on all transmissions of type p. If there are c_p such transmissions in total, we can enumerate them in chronological order, and denote them by $p_1, \ldots, p_{c_p}$.

The solution approach is based on constructing a directed generalized graph (ggraph) representation of the knowledge about the event order. The log entries constitute the nodes of the ggraph. There are *two* kinds of edges: "earlier-than" (E) edges and "earlier-than-or-equal-to" (EEQ) edges. We initialize the ggraph by including E edges between each pair of consecutive log entries within each log file. For each reception $x \in R$, we furthermore add two EEQ edges. Let $p = P(x)$, then x must stem from one of the transmissions $p_1, \ldots, p_{c_p}$. We may thus add an EEQ edge from p_1 to x (because x can impossibly have happened before p_1), and an EEQ edge from x to p_{c_p} (because x cannot have happened after p_{c_p}). Each EEQ edge leads from a send to a receive event or vice versa. Each receive event has exactly one incoming EEQ edge and exactly one outgoing EEQ edge. In the course of the algorithm, the EEQ edges will be iteratively replaced, whenever it can be excluded that the receive event at one end belongs to the send event at the other end.

Such an exclusion is possible if, for an EEQ edge (a, b), there is a path from a to b in the ggraph that traverses at least one E edge. If this is the case, a must have happened *strictly* before b; due to property (2) in Sec. 2, a and b can thus impossibly be a send-receive pair. Let $p = P(a) = P(b)$. First consider the case where a is the send event and b is the receive event. Then, $a = p_i$ for some $i \in \{1, \ldots, c_p\}$. If p_i cannot be the source of b, then b must stem from p_{i+1} or a later event. We thus replace (a, b) by an EEQ edge originating from p_{i+1}, i. e., by (p_{i+1}, b). Analogously, if $b = p_i$ is the send event, then a must belong to p_{i-1} or an earlier event, so we replace (a, p_i) by (a, p_{i-1}).

This can be iterated until no further "invalid" EEQ edges are detected. For each receive event x, let α_x denote the origin of x's incoming EEQ edge, and ω_x the endpoint of x's outgoing EEQ edge after termination. It is easy to see that x must necessarily originate from one of the send events "between" α_x and ω_x (inclusive): all other send events of type $P(x)$ have been excluded as impossible in one of the substitution steps. The algorithm has therefore narrowed down the send candidate set. Actually, in all our experiments it terminated with *correct* (that is, minimal) send candidate sets for all receive events. However, so far, we did not succeed to prove this property in the general case.

If n is the total number of log entries, the number of edge substitution steps for each receive event is obviously bounded by $O(n)$. Consequently, the total number of substitution steps until termination is in $O(n^2)$. After each substitution, we must re-check all EEQ edges whether they are still valid. At any time, there are less than $2n$ EEQ edges in the ggraph, so $O(n^3)$ checks are necessary in the worst case. Both checking and substitutions can be implemented efficiently, so the computational complexity of the algorithm is polynomial. A more in-depth analysis is left for future work.

7. CONCLUSION

We introduced the send-receive correlation problem, which appears in network log analysis. It is relevant, e. g., in the context of offline time synchronization. We formalized the problem and outlined an approach to algorithmically solve it. In the future, we intend to investigate it further, including several variants with relaxed or tightened assumptions.

Acknowledgments

The authors are grateful to a number of people for helpful discussions and feedback, most notably (in alphabetical order) Frank Gurski, Florian Jarre, Daniel Marks, Alexander Marold, and Egon Wanke.

8. REFERENCES

[1] Y.-C. Cheng, J. Bellardo, P. Benkö, A. C. Snoeren, G. M. Voelker, and S. Savage. Jigsaw: Solving the puzzle of enterprise 802.11 analysis. In *SIGCOMM '06*, Sept. 2006.

[2] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00*, pages 243–254, Aug. 2000.

[3] L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[4] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Analyzing the MAC-level behavior of wireless networks in the wild. In *SIGCOMM '06*, Sept. 2006.

[5] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), Nov. 1982.

[6] 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*, 17(2):431–444, Apr. 2009.