FSM BASED INFERENCE OF MISSING PACKET IN WLAN

Kapil Mestry[1], Rupesh Shirodkar[2]
BE computer ,SSPM's college of Engineering (SSPMCOE), Kankavli, India

Abstract
Wireless network becoming a crucial part of ICT based organization. The network communication traces are useful in user behavior analysis, security, resource utilization, network management. These traces are basis for the decision making systems, big data analysis. More accuracy will be achieve with more complete datasets. We are proposing FSM based inferring system for missing packets in infrastructure based WLAN.
Keywords:missingpackets;WLAN;Packet_Trace; styling; insert (key words)

I. INTRODUCTION

Nowadays wireless network becoming important part in our life such as smartphones,tabs were used by people for communication, sharing information on social networking sites also people use the network for commercial purpose like online shopping.This gives raise to develop more robust and secure ccommunication protocol.Also these wireless communication can be traced and used for user bahaviour analysis,for Monitoring the network activity,for network management.There are different tools are available such as wireshark,tshark,ns tool used in the communication protocol analysis and research. The packet traces were captured across different access points by using monitors.This traces will be used for various purposes.The accuracy of the traces neccessary for detailed MAC level behaviour of operational wireless network.[3]The incomplete views from multiple monitors will be merged for detail analysis.There are some packets which are not captured by any monitor which affects the performance of applications using the traces. We are implementing a framework for reconstruct the packets that are missing from the trace. There are different approaches to deal with this problem.AP logs provide information on transmission and reception of AP, but not those of client.One or more host in the BSS record the attributes of all trasmission that they observe.Which will be collected to form the complete trace of communication with some missing packets due to packet drop cause by collision or weak signal.It is not easy to find how much information is missing. Our goal is to estimate the missing packet from merged trace collected across different
Monitors.The second goal is to develop inference procedure based on formal lenguage model to determine wheather each packet was recieved by its destination and add the packets that are missing from merged trace.

II. RATIONALE AND SIGNIFICANCE OF THE STUDY

The inference engine uses the information in the packet that the monitor captures to infer the packets that are not capture. We use regular language as our choice of formal language since They are recognizable by finite state automata which have better implementation. FA also provides efficient way to enhance traditional language recognition in a way that allows sentence reconstruction from partial information. To infer missing information we scan the trace and process each packet.

III. PROBLEM DEFINATION

After going through the literature and discussion we have formulated the problem statement, scope and objectives of our project as follows:

A. *Problem Statement*

Traces collected by different monitors or sensors for different purpose such as:

- Research
- User moment tracking

It is required that the traces should contain all the packets for every conversation to be infer. Tools such as wireshark will be used for capturing the packets will skip some of the packets.

Our aim is to infer the complete trace.

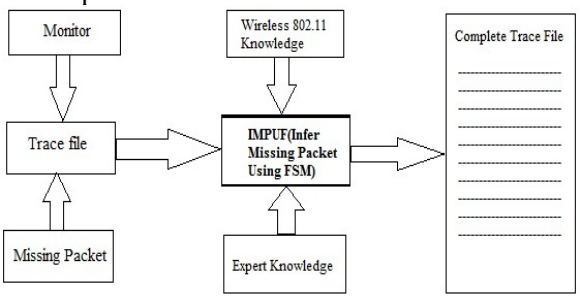
B. *Scope*

- The packet traces consist of only infrastructure based communication.
- The system works in offline mode.
- Human intelligence may be required for some critical communication path.
- The packet trace should contain enough no of packets between two nodes to predict missing packets.

IV. PROPOSED SYSTEM

A. *System Architecture*

We define a system named IMPUF(Infer Missing Packet using FSM).It is a non-intrusive tool that builds on passive monitoring to support detailed MAC-level analysis of operational 802.11 wireless networks. It uses an engine based on formal language techniques to infer packets that were missed by all monitors as well as infer which packets were received by their destinations. We use traces to evaluate our techniques.



To resolve the problem defined in the problem definition we define the architecture of our system. Which takes input as a trace file that is captured by monitors consist of packets that are Communication between two end that is single source and destination. In such a trace file some packets are missing which are transferred during communication between two ends. Our task is to find out such packets or generates such packets based on formal language technique for that we use FSM (Finite State Automata).

B. *Working of System:*

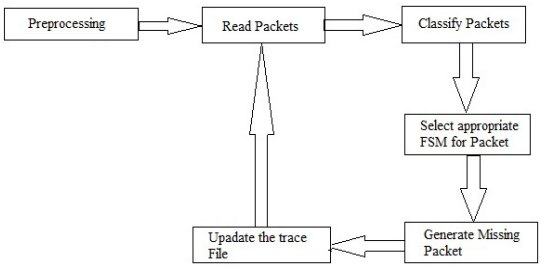


Fig. (IMPUF) Inference of missing packet using FSM

- Preprocessing Phase:

Input: the trace file captured by the monitors.

Output: A database table having details of the packet plus additional marker packets. The rows will be sorted based on time and clubbed together representing continuous conversation between single source and destination.

- Read Packets:

Input: It reads the packet in the conversation corresponding to selected pair of source and destination.

Output:

- Classify Packets:

Input: Packet trace captured by monitor.

Output: Classify the packets according to the type of packets present in packet trace

- Select appropriate FSM:

Input: Classified packet trace. We map packets to symbols of the language of FSM. Additionally, we identify the conversation of the packet based on its source and destination. Non-unicast packets are considered conversations of a single packet.

Output: Appropriate FSM use for finding missing packets in the trace.

- Generate Missing Packet:

Input: Identified missing packet by FSM.

Output: Generate or build the missing packet in the trace.

- Update the missing packet:

Update the trace with generated missing packet in the trace.

V. METHODOLOGIES AND TECHNIQUES

A. *Formal Language Approach*

We cast the inference problem as a language recognition task. Sentences in the language represent legal sequences of packets ex- changed by two endpoints that follow the protocol. We call these packet exchanges conversations and define them at the granularity of logical 802.11 operations (e.g., all packets involved in an association attempt, or an exchange involving

RTS, CTS, DATA, and ACKs to successfully convey a single data packet). Although longer conversations can be defined (e.g., association must precede data transmission) to enable a slightly larger set of inferences, the practical benefit of doing so is tiny; the additional inferences are about relatively rare events. We view the input trace as interleaved partial sentences from the language. The interleaving stems from overlapping conversations between distinct endpoint pairs. A similar view of packet traces is taken in the context of passive testing of protocol implementations. Our goal is different, however: to find valid sentences in the language that account for what is observed in the input trace. Thus, we do not simply ask "Is this sentence in the language?" Rather, we presume that there was a sentence in the language for which we see only some of the symbols and ask what complete sentence it was likely to have been. We use regular languages as our choice of the formal language because they are recognizable by finite state machines (FSMs) which have efficient implementations. FSMs also afford an efficient way to extend traditional language recognition in a way that allows sentence reconstruction from partial information, as described below. Our short conversations can be easily described using FSMs.

B. *Processing the Trace*

Assume that the FSM (and so the language) for our protocol has been defined. To infer missing information using it, we scan the trace and process each packet as follows:

1) Classify:

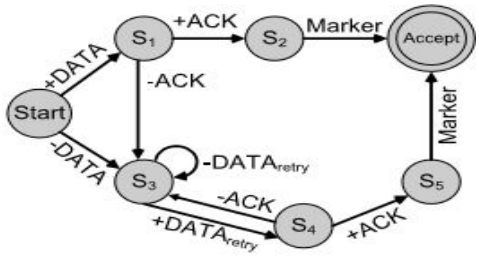
We map packets to symbols of the language based primarily on their type. We also use the values of the retry bit and the fragment number field in forming symbols, which provides some additional leverage in making inferences, at the cost of a somewhat larger symbol set and FSM. Additionally, we identify the conversation of the packet based on its source and destination. For packets without the source field (ACKs and CTSs), we deduce the source from earlier packets. Non-unicast packets are considered conversations of a single packet.

2) Generate Marker:

Our language contains an artificial symbol, which we call the marker. We introduce a marker if the currently scanned packet indicates that an ongoing conversation has ended. This occurs under one of the following conditions. First, the sequence number field signals a new conversation between the endpoints. Second, for non-AP nodes, the other endpoint of the current packet is different from the earlier one; only APs can have multiple simultaneous conversations. Third, there is no legal transition in the FSM for the current symbol; if nodes correctly implement the 802.11 protocol, our FSM construction (described below) ensures that there is always a transition for packets in the current conversation. Fourth, a timeout interval has passed since the last seen activity for the conversation.

3) *Take FSM Step*:

If a marker was generated, first take a step in the FSM based on the marker. By construction, this causes a transition to the accept state, closing the current conversation and placing the FSM in the start state. The path taken from the start to the accept state reveals information missing from the trace, as explained shortly. Now take a step in the FSM based on the symbol for the current packet.



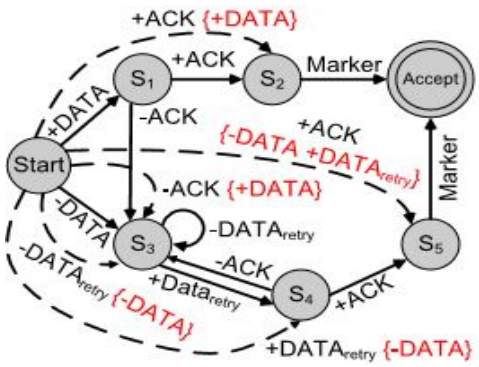
An FSM for our simplified example. A '+' indicates that the packet was received by its destination, and a '−' indicates that the packet was lost.

While the first two steps involve some 802.11-specific decisions, the third step is entirely independent of the protocol being analyzed. Key to this process is the construction of the FSM, which re- quires elaboration. We cannot simply use an FSM corresponding to the protocol because packets (i.e., sentence symbols) are missing from the trace and because we want to use the FSM to estimate which packets were received by their destination. We extend traditional FSM matching to address these issues. We explain our method in the context of a simplified version of 802.11 data exchange conversations in which there are no fragments,

and instead of quitting after a configured number of at- tempts, nodes retransmit the data packet until they receive an ACK. An FSM for this simplified version is shown in above figure.

1) *Inferring Missing Packets:*

We now explain how we infer packets that are missing from the trace. With missing packets, there may be no legal transition for the current symbol.



The same FSM after augmentation of the Start state (only).The dashed edges are the augmented ones and the symbols in braces are their annotations.

For instance, in our simplified example, if the first packet encountered for a conversation is a DATA retry, there is no legal transition out of the start state. Our solution is to augment the FSM with additional edges. Abstractly, for each pair of states (Si,Sj) 6 = (Start, Accept), we add an edge from Si to Sj for each distinct trail(or, a path with no repeated edges) from Si to Sj , labeling it with the final symbol of the trail. We annotate each augmented edge with the traversed trail's prefix, i.e. the path without the final symbol. The annotation represents packets that must be missing if the edge is traversed to reach the accept state. Example augmented edge scan be seen on Figure 4. For instance, the edge between Start and S4 can be taken upon observing a DATA retry and its annotation indicates a missing DATA packet (which was lost). We move non-deterministically in the augmented FSM until the accept state is reached. At this point, there may be multiple paths from Start to Accept, all of them consistent with the captured packets. To select, we assign weights to paths and select the lowest weight one. The weight of a path reflects the number of packets that it indicates as missing and the rarity of those packets types. Specifically, it is the sum of the weights of its edges. Unaugmented edges, which correspond to captured packets, have zero weight. The weight of an augmented edge is the sum of the weights of the symbols in the annotation. Symbol weights are inversely proportional to their frequencies in the trace. (We find that our inferences are similar even when we use the logs of these values, which translates the decision to the minimum product of the inverse frequencies, rather than their sum.) This weighting method prefers the shorter of two paths when the symbols of one are a subset of the other, thus producing conservative estimates of missing packets. When the path weight function is a linear operator, as in our case, a straightforward optimization simplifies FSM construction, without impacting results. If there are multiple trails from Si to Sj ending with the same symbol, only the lowest weight one needs to be considered. The Figure 4 shows the FSM for our example after the Start state (only) has been augmented using this optimization. As a final step when the accept state is reached, we synthesize any missing packets along the selected path. We cannot always infer the exact properties of a missing packet but can often do so. Properties that are relevant for MAC-level analysis include packet size and transmission time and rate, and which of these we can infer depends on the details of the 802.11 protocol. The size of certain packet types, such as ACK, RTS and CTS, is fixed. For others, such as DATA packets, the size can be inferred if a retransmission of the packet is observed. The transmission time of a missing packet can be inferred if there exists a captured packet relative to which it has a fixed spacing; for instance, the transmission time of a DATA packet can be inferred from that of the corresponding ACK. The trans- mission rate of certain packet types, such as PROBE REQUEST, is usually fixed for a client, and for certain other types, such as ACK, it depends on the rate of the previous, incoming packet (i.e., DATA). However, the rate of missing DATA packets cannot be inferred unless the rate adaptation behavior of the sender is known.[2]

VI. IMPLEMENTATION DETAILS

We have basically two modules implementing the proposed system. First preprocessing in which we upload the trace to database and generate the marker. Second FSM module which reads the packets from database and steps though the FSM to generate the missing

packets. For FSM we have created *tt_table* for storing the state transition diagram and *ot_table* for storing the output symbols for transition from one state to another state in FSM. We have tried for association phase for which we have 8 states with 9 input symbols.

A. Attributes of wireless trace

Traces vary in the information they include. Some traces have timestamps precise to nanoseconds, others only to milliseconds; not all traces record 802.11 acknowledgments; to maintain users' anonymity, few researchers release full payloads, and so on [13, 15]. The following data are available in all 802.11 CRAWDAD traces[5]; we assume them as the core data that are likely to be available in future wireless traces:

1. All types of data packets.
2. All types of management packets including beacons, probe requests, and probe responses.
3. Full 802.11 header in all captured packets, including source and destination addresses (possibly anonymized), Sequence number, retransmission bit, type, and subtype. Beacon packets[1] also have timestamps applied By the AP.
4. Monitor's timestamp (set by the kernel or possibly the device).

VII. PERFORMANCE EVALUATION

A. Completeness of the trace

We estimate the completeness of the trace while using Ct which is defined as number of packets with sequence number changed per node. [4]

$$Ct=\sum \text{sequence\_no\_Change of node i}$$

B. Number of missing packets generated

Mt=no_of_output symbols inserted in the trace during FSM transition.

Performance can be evaluated by using formula

$$Efficiency=Mt/Ct;$$

REFERENCES

[1] Pablo Brenner, “BREEZECOM wireless communication” ,Technical tutorial on the IEEE 802.11 Standard"

[2] R. Mahajan, M. Rodrig, D.Wetherall and J. Zahorjan, “Analyzing MAC level behavior of wireless networks in the wild.", in SIGCOMM’06

[3] J.Yang, YChen, W.Trappe,J.Cheng, "Detection and localization of multiple spoofing attackers in wireless networks.", IEEE Transactions on parallel and Distributed Systems, Vol. 24,No.1,Jan. 2013

[4] Aaron Schulman” On the Fidelity of 802.11 Packet Traces” Master's Scholarly Paper by Neil Spring Department of Computer Science, University of Maryland, College Park

[5] CRAWDAD data set Downloaded from http://crawdad.cs.dartmouth.edu.