Real Traffic Replay over WLAN with Environment Emulation

Chia-Yu Ku
Department of Computer Science
National Chiao Tung University
Taiwan
cyku@cs.nctu.edu.tw

Ying-Dar Lin
Department of Computer Science
National Chiao Tung University
Taiwan
ydlin@cs.nctu.edu.tw

Yuan-Cheng Lai
Department of Information Management
National Taiwan University of Science and Technology
Taiwan
laiyc@cs.ntust.edu.tw

Pei-Hsuan Li
Department of Computer Science
National Chiao Tung University
Taiwan
estelle.cs98g@nctu.edu.tw

Kate Ching-Ju Lin
Research Center for IT Innovation
Academia Sinica
Taiwan
katelin@citi.sinica.edu.tw

***Abstract*—Real traffic replay is one of the solutions used to test network devices over complicated scenarios. Packet traces captured in a real environment hold more details than any mathematical models. However, the lack of packet-replay control and environment emulation might highly affect traffic behaviors, especially in wireless networks. Real traffic replay in wireless networks requires packet-replay control to manage the interactions with the device under test (DUT), and coordinately reproduces environment effects, such as fading, noise, and interference. In this work, we propose a method, called Event-driven Automata-synchronized Replay (EAR), to address real traffic replay over WLAN. EAR transforms the captured packet trace into a sequence of events that follow the IEEE 802.11 protocol. The three-level automata are applied to achieve packet-replay control and synchronize the environment effects in traffic replay with the packets and signals captured in a real environment. We propose a quantitative metric, called the event reproduction ratio (ERR), to evaluate the effectiveness of traffic replay. Our software implementation on the Linux-based system demonstrates that EAR achieves the ERR of 95.9% and 92.45% over the DUT-dependent traffic and fading environments, respectively. Under the same condition, the straight-forward replay can only produce the ERR of 20.6% and 0%, respectively.**

Keywords- traffic replay, real traffic, WLAN, wireless network, environment emulation

I. INTRODUCTION

Traffic replay is one of the solutions used to test network products over complicated scenarios. Based on the methodologies of traffic generation, traffic replay can be classified into *trace-based* and *model-based* technologies. Model-based traffic replay utilizes statistical methodologies to build a mathematical model of traffic patterns or environment conditions. The pre-defined parameters need to be provided for configuring the model. However, the complicated details of network operations might be neglected by inaccurate modeling. On the other hand, trace-based traffic replay uses the packet traces and environment information recorded in a real environment; hence, it contains most of the details in a real environment. The more details the trace retains, the more phenomena the traffic replay can reproduce.

Real traffic replay is widely applied in wired networks over the last decade. Cheng [1] studied the stateful issue of traffic replay in wired networks. The packet exchanges between the device under test (DUT) and the device used to perform traffic replay should be precisely controlled based on the protocol specification. For example, consider a scenario where an access point (AP) is our DUT, which communicates with a station (STA). Traffic replay should not transmit the Association Request before receiving the Authentication from the DUT. However, due to variant wireless environments, traffic replay in wireless networks is more challenging. Because wireless medium is open and shared, transmissions may suffer packet errors caused by environment effects, such as fading, noise, and interference. Therefore, traffic replay should consider the impact of those environment effects. Even though the environment effects could be reproduced, poor coordination between packet-replay control and environment emulation can also result in inconsistent environment effects with the reproduced packets. As a result, traffic replay in wireless networks must consider packet-replay control and environment emulation jointly.

A number of studies investigated the issues of traffic generation and environment emulation. Traffic generation discussed in [2-4] considered the key characteristics of traffic, such as the packet size, to reproduce packets. Moreover, packets are sent according to timestamps under per-session control of the transport and application layers. On the other hand, environment emulation studied in [5-8] simulated the transmission failure probability according to the signal strength and channel busy probability. However, all the above approaches did not consider traffic generation and environment emulation jointly. Pradipta [9] and Judd [10] attenuated the signal strength of the reproduced packets according to the information of the recorded time-domain signals. Nevertheless, their approaches cannot handle bi-directional transmissions due to the lack of interactions with the DUT.

To the best of our knowledge, few previous studies devoted to traffic replay that takes both packet-replay control and environment emulation into account. In this paper, we propose a method called *event-driven automata-synchronized replay* (*EAR*). EAR defines a series of events that follow the IEEE 802.11 protocol [11] to describe traffic behaviors, and applies automata to control the reproduced packets and environment effects. We then use a quantitative metric, called the *event reproduction ratio* (*ERR*), to evaluate how well the reproduced events in traffic replay approximate the behaviors of real traffic. We implement EAR as software in Linux-based systems and empirically evaluate its performance in terms of the ERR in a WLAN testbed.

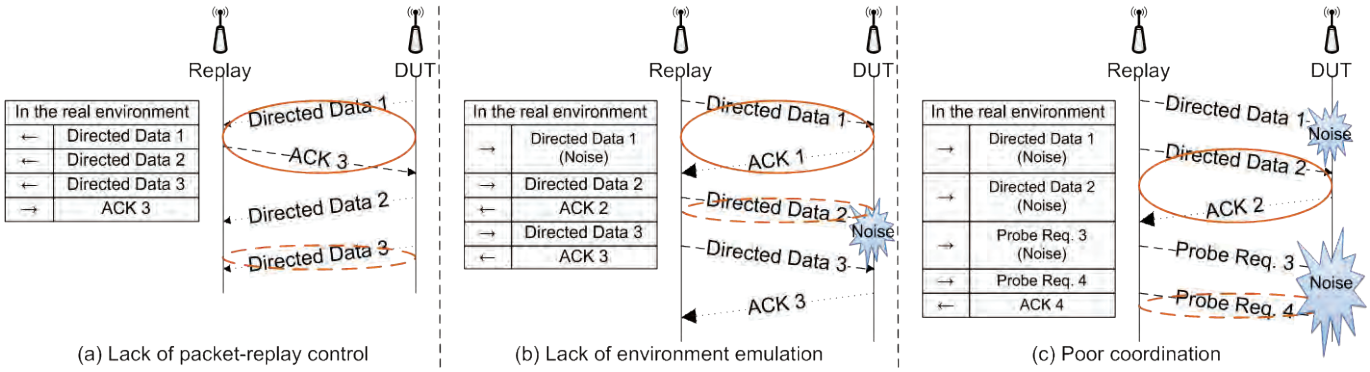


Fig. 1 Examples of FN/FP cases in real traffic replay

The rest of this paper is organized as follows. Section II shows the design issues and problem statement. The design details of EAR are described in Section III. The performance evaluation of EAR is investigated in Section IV. Finally, we conclude this paper in Section V.

II. Problem Statement

This section explains the issues of real traffic replay over WLAN and formulates the problem statements. We provide some illustrating examples where the traffic replay is inconsistent with the real environment. To analyze complex traffic behaviors, EAR transforms packets into a series of events, and defines the ERR to quantify how the events reproduced in traffic replay match those in the real environment.

A. Design Issues

Compared to the packet traces recorded in a real environment, the reproduced traffic might be *false negative* (*FN*) and *false positive* (*FP*). FN indicates that a packet exchange succeeds in a real environment, but fails in traffic replay. On the contrary, FP indicates that a packet exchange fails in a real environment, but succeeds in traffic replay. The lack of packet-replay control, environment emulation, or coordination between them could result in FN and FP cases.

Fig. 1(a) shows an example where traffic replay reproduces ACK 3 after Directed Data 1 due to the lack of packet-replay control. In this example, traffic replay should reproduce ACK 3 after receiving three Directed Data from DUT. However, Replay transmits ACK 3 immediately after Directed Data 1 regardless of the interactions with DUT. Therefore, the reproduced packet exchange does not match correctly with that in the real environment. In this case, Directed Data 1 and 3 are FP and FN, respectively. On the other hand, transmissions in wireless networks are prone to bit errors in variant environments. In Fig. 1(b), Directed Data 1 is FP because the noise that disturbs Directed Data 1 is not reproduced. As a result, the packet error rate (PER) in the traffic replay is lower than the PER in the real environment. In contrast, Directed Data 2 is FN because it is transmitted correctly in the real environment, but is disturbed by an unexpected noise in the traffic replay.

Even though the packet-replay control and environment emulation is applied, the poor coordination between them can also result in FN or FP cases. Fig. 1(c) shows that the duration of the reproduced noise does not match the duration of the reproduced packet exchange. Timing of transmissions could be different in a real environment and traffic replay due to medium access overheads and behaviors of DUT. Because the duration of the recorded noise in the real environment is not consistent with the duration of the packet exchange in the traffic replay, Directed Data 2 and Probe Req. 4 become FP and FN, respectively. The above examples illustrate that the lack of packet-replay control or environment emulation highly affects the traffic behaviors. Therefore, both of packet-replay control and environment emulation should be reproduced coordinately.

B. Problem Formulation

To deal with complex traffic behaviors, we define a formal statement to describe packet exchanges. Because a single packet represents insignificant behaviors in the view of protocols, this paper defines a series of *events* according to IEEE 802.11 to represent packet exchanges. For instance, RTS/CTS followed by the exchange of Directed Data and ACK can be treated as a single event. Thus, a given packet trace can be transformed into a sequence of events. Let E_{Real} denote the sequence of events in a real environment; let E_{Replay} denote the sequence of events reproduced by traffic replay. The i^{th} event of E_{Real} is named $E_{Real}(i)$, which is set to 0 if the i^{th} event in a real environment succeeds and set to 1, otherwise. Similarly, the value $E_{Replay}(i)$ is set to 1 or 0 depending on whether the i^{th} event of E_{Replay} is successful or failed, respectively.

Given the captured packet trace, EAR classifies the packets into a sequence of events, i.e., E_{Real}. According to E_{Real}, EAR reproduces each captured packet and synchronizes the reproduced packets with the environment effects. Next, EAR transforms the reproduced packets into a sequence of events, i.e., E_{Replay}. To reproduce the traffic behaviors, EAR tries to make E_{Replay} as consistent as possible with E_{Real}. In another words, the events in the traffic replay perfectly match the events in the real environment, $E_{Real}(i) = E_{Replay}(i), \forall i$. To quantify the approximation level of the traffic replay, EAR calculates the *event reproduction ratio* (*ERR*) by

$$\frac{\sum_{i=1}^{n} \neg(E_{Replay}(i) \oplus E_{Real}(i))}{n} \times 100\%, \quad (1)$$

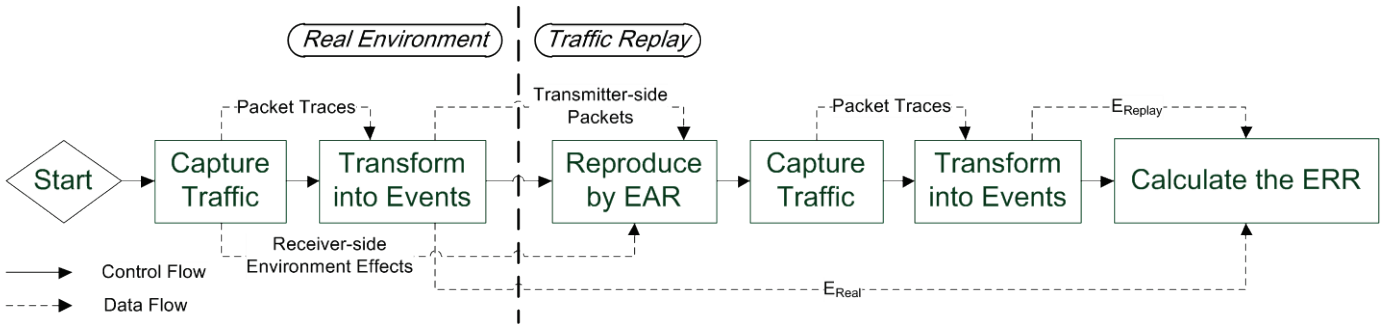


Fig. 2 Flowchart of EAR

where n is the number of events in E_{Real}. Ideally, EAR aims at maximizing the ERR. However, even though the environment effects can be perfectly emulated, the probability that $E_{Real}(i)$ and $E_{Replay}(i)$ are equal is still less than 1 because $E_{Real}(i)$ and $E_{Replay}(i)$ could be 0 or 1 based on the PER. Therefore, the ERR might not equal 100%. We demonstrate this phenomenon in Section IV.C.

III. Event-Driven Automata-Synchronized Replay (EAR)

This section details the proposed EAR, a traffic replay solution that addresses the issues described in Section II. EAR transforms a captured packet trace into events, and then reproduces each event synchronized with the environment effects based on the automaton. Finally, the ERR is calculated based on Eq. (1).

A. Overview

The traffic captured in the real environment consists of a sequence of packets. Because reproducing traffic in a packet-by-packet manner results in insignificant traffic behaviors in the view of protocols. Therefore, instead of reproducing packets, EAR classifies packets into a sequence of events, and addresses traffic replay and evaluation according to the events. Fig. 2 shows the flowchart of EAR.

Because of bidirectional transmissions, the DUT can play the role of a transmitter or a receiver. For example, the AP (DUT) in Fig. 3 is the receiver of the Data packets, but the transmitter of the ACK packets. In variant wireless environments, the packets and environment effects seen by the transmitter and receiver could be different. A transmitter cannot see the environment effect of each packet caused by the wireless channel. On the other hand, a receiver might not get correct packets because some packets are corrupted due to the fading environment. Therefore, EAR extracts E_{Real} and E_{Replay} from the *receiver-side packets* that are captured at receivers. On the contrary, EAR needs to reproduce the *transmitter-side packets* that are captured at transmitters, and the corresponding *receiver-side environment effects* that are measured at receivers. Fig. 3 illustrates the events of E_{Real} and the corresponding trace for traffic replay.

E_{Real} can be extracted from the receiver-side packets, i.e., the black left arrows at Monitor 1 and the white right arrows at Monitor 2. On the other hand, the traffic replay reproduces the transmitter-side packets, i.e., the black right arrows at Monitor

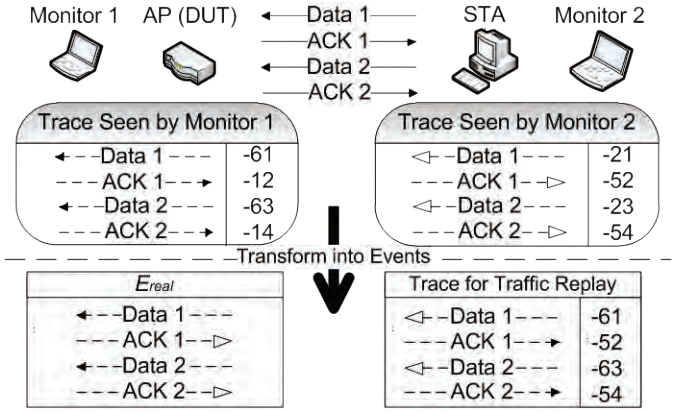


Fig. 3 An example of E_{Real} and the corresponding reproduced traffic replay

1 and white left arrows at Monitor 2. EAR also adds the receiver-side environment effects measured by the monitors. In this example, the received signal strength is -52 and -54 dBm for the black right arrows, and -61 and -63 dBm for the white left arrows, respectively.

B. Reproduce by EAR

1) Packet-Replay Control

To keep correct interactions with the DUT, EAR reproduces traffic based on the three-level automata, as shown in Fig. 4. Fig. 4(a) depicts the *level-0* automaton that maintains the *station states* of a STA at both of the STA and the AP. The valid packets in each state are classified into Class 1, 2, and 3 [11] according to the Authentication and Association statuses. To make all the reproduced packets acceptable to the DUT, EAR keeps the same station state as that in the real environment. When detecting a state transition caused by unexpected behaviors of the DUT, EAR applies the Authentication or Association procedure to roll back the station state of the DUT. For example, when the DUT sends a De-authentication or a Disassociation, EAR can detect the change of station states, and then perform the Authentication or Association procedure to re-authenticate or re-associate with the DUT.

The *level-2* automata address the *atomic events* that should not be interrupted by other events. Atomic events can be mapped to the operation sequences defined in IEEE 802.11. The state transitions of the level-2 automata can only be triggered by packet exchanges. In Fig. 4(c), except the *Broadcast* automaton that does not interact with the DUT, in

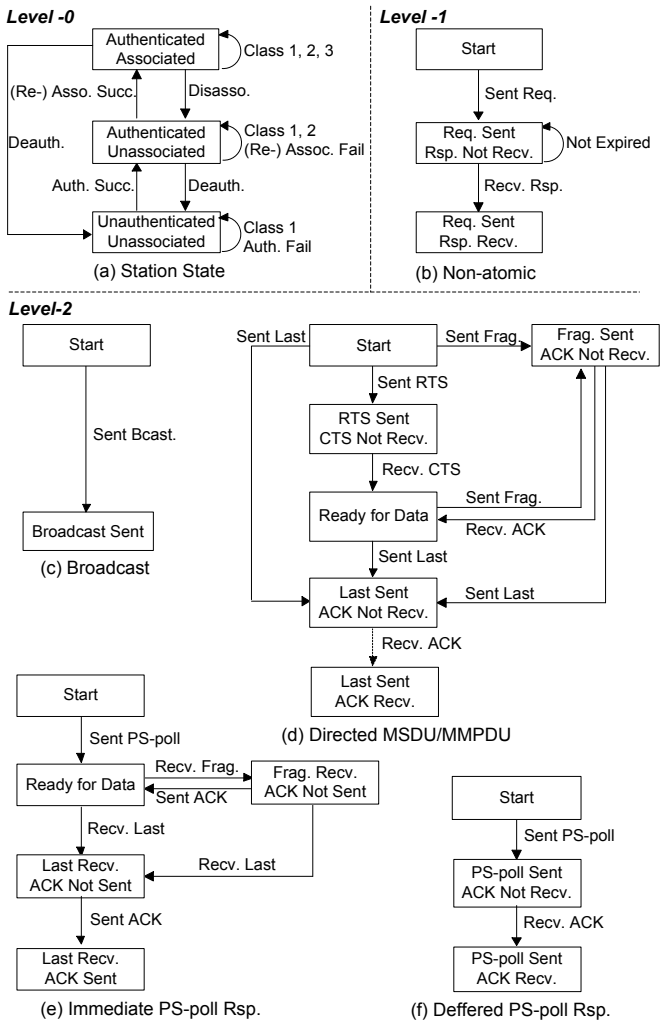


Fig. 4 Three-level automata

all the other level-2 automata, EAR should transmit a packet after the previous packet has been received from the DUT. The *Directed MSDU or MMPDU* automaton addresses the unicast data or management frames. The *Immediate PS-poll Rsp.* and *Deferred PS-poll Rsp.* automata handle retrieving the buffered data during the power-saving mode [11]. In the figures, *Frag.* denotes a fragment of data, and *Last* indicates the last fragment of data.

The *level-1* automaton deals with the *non-atomic* events, which allow other events to occur between the request and response. The state transitions of the level-1 automaton are triggered by atomic events. For example, after sending the Association Request, EAR also allows redundant Probe Requests and Responses before receiving the Association Response from the DUT. In this case, the association procedure is still thought of completing correctly. Table 1 summarizes the relationships among the automata and events.

2) Environment Emulation

When EAR reproduces the captured packets, the corresponding environment effects should also be generated. Specifically, each packet reproduced in traffic replay should

TABLE 1 RELATIONSHIPS BETWEEN EVENTS AND THREE-LEVEL AUTOMATA

Level	***Automata***	***Event***
0	Station state	Station state
1	Non-atomic	Auth.
		Assoc.
		Re-assoc.
		Active Scan
2	Broadcast	Broadcast Data
		Broadcast Probe Req.
		Beacon
		Interference
	Directed MSDU/MMPDU	Unicast Data
		Unicast Probe Req. / Probe Rsp.
		Auth. Req. / Auth. Rsp.
		Assoc. Req. / Assoc. Rsp.
		Re-assoc. Req. / Re-assoc .Rsp.
		De-auth.
		Disassoc.
	Immediate PS-poll Rsp.	Immediate PS-poll
	Deferred PS-poll Rsp.	Deferred PS-poll

be synchronized with the environment effect it suffers in the real environment. Thus, EAR synchronizes the reproduced packets and environment effects by the level-2 automata. Because the transitions of the level-2 automata are triggered by packet exchanges, EAR adjusts the environment factors at each state transition.

In wireless communications, several environment factors, such as fading, noise, and interference, may increase the bit error rate (BER). Fading represents that the signal strength of the transmitted packet decreases due to path loss or shadowing. Noise is a random disturbance signal that cannot be recognized by the wireless communication system. Interference indicates the traffic originated from other devices that share the same medium. To emulate the above key environment factors, EAR uses an attenuator to adjust the signal strength according to the RSSI [11] measured in the real environment. Moreover, EAR uses a vector signal generator to generate AWGN according to the recorded noise strength. Finally, EAR emulates the interference by reproducing the captured packets that do not terminated at the DUT. Because these interference packets do not interact with the DUT, EAR treats them as a special event controlled by the *Broadcast* automaton.

Fig. 5 illustrates an example of synchronizing the environment effects with the packet-replay control. When EAR transmits the reproduced Req. to the DUT, the signal and noise strength is set to -40 and -71 dBm, respectively. Because the channel conditions measured in the real environment could change, EAR changes the signal and noise strength to -42 and -73 dBm, respectively, until the DUT replies Rsp.

IV. EVALUATION

We evaluate the performance of EAR in terms of the ERR. The experiments examine the packet-replay control and environment emulation separately. Moreover, a case study is provided to observe the relationships between the ERRs and common upper-layer protocols.

A. Evaluation Environment

The experiment environment is shown in Fig. 6. The

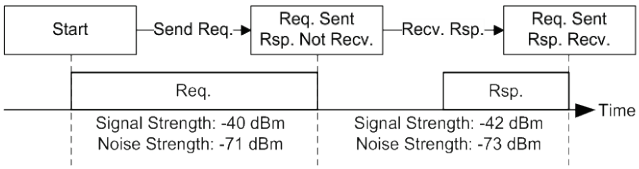


Fig. 5 An example of synchronizing environment effects

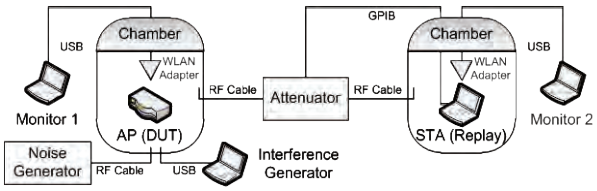
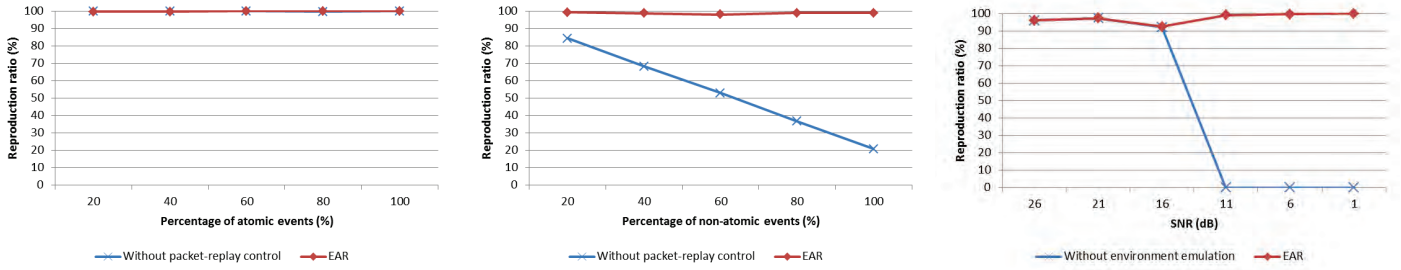


Fig. 6 Evaluation testbed



(a) Different proportions of atomic events (b) Different proportions of non-atomic events (c) Different signal strength

Fig. 7 ERRs of EAR

testbed consists of an AP, which plays the role of the DUT, and a STA, plays the role of EAR. The AP and STA are put in two different chambers, which are isolated from the uncontrollable open environments. To capture the packets and environment effects received by both of STA and AP, two monitors are connected to the WLAN adapters through the USB interface. Both chambers are connected to an attenuator through a RF cable to adjust the signal strength of the reproduced packets. At the DUT side, a vector signal generator and a laptop equipped with a WLAN adapter generate the noise and interference, respectively. We put all the devices at the fixed positions and calibrate them carefully before the experiments. Moreover, the atomic event utilizes a simple unicast data event that sends a data frame and waits for an ACK. Similarly, the non-atomic event utilizes an event consists of the requests and responses of management frames, such as Authentication and Association. The other events in the experiments are data broadcast events that do not require packet-replay control.

B. *Packet-replay Control*

We first examine the impact of packet-replay control. To eliminate the impact of environment effects, we set the signal strength to a high enough value, i.e., -64 dBm in our testbed, to minimize packet errors. Fig. 7(a) and Fig. 7(b) show the ERRs over different proportions of atomic and non-atomic events. In Fig. 7(a), EAR can achieve the ERRs of close to 100%. To guarantee the atomic property, the IEEE 802.11 protocol requires two constitutive packets to be separated by the Short InterFrame Spacing (SIFS). Therefore, the atomic events are timing-sensitive. However, EAR is implemented by Libpcap [12], which is a user-space library for communications between a user-space program and a network interface card (NIC). The packets generated by EAR in the user space are too slow to interrupt the SIFS. Even if EAR is implemented in the WLAN driver, it could not transmit packets with an interval shorter than the SIFS because the WLAN chipsets do not allow this behavior in order to follow the 802.11 standard. Therefore, no FN or FP case for atomic events would happen during traffic replay.

Fig. 7(b) shows that EAR can achieve the ERRs higher than 95.9%. However, the ERR of the traffic replay without packet-replay control decreases to 20.68% when the proportion of the non-atomic events increases. This is because the non-atomic events could be interrupted by other events. Hence, the traffic replay without packet-replay control may result in an out-of-order sequence of events. For example, the Association procedure may start before the Authentication procedure finishes. However, EAR uses the automata to maintain the expiration time and correct packet sequence. Therefore, EAR can reproduce the events correctly.

C. *Environment Emulation*

To observe how environment effects affect the ERR, this experiment applies packet-replay control and adjusts the signal strength of each reproduced packet. The noise strength in the chamber is treated as a constant and the mean of the measured noise strength is -85 dBm. Fig. 7(c) exhibits the ERRs under different signal strength. The ERRs in all the experiments are higher than 92.45%. When the signal strength is high, both of EAR and the traffic replay without environment emulation can achieve a high ERR because of few packet errors they suffer. Nevertheless, the PER increases when the signal strength decreases. As a result, the ERR of the traffic replay without the environment emulation also decreases under the low signal strength because the reproduced packets might not suffer packet errors. On the contrary, EAR emulates the environment effects and introduces the packet errors that match the real environment. Notice that the ERR at -69 dBm is lower than the others. The reason is that, when the signal strength is extremely high or low, the PER is close to 0 or 1. By contrast, the PER is not an extreme value when the signal strength is medium. Because each event in traffic replay is reproduced with a probability based on the PER measured in a real environment, E_{Real} might not match E_{Replay} when the PER is not close to 0 or 1.

D. *Case Study*

These experiments investigate the network-layer (ICMP)

TABLE 2 CASE STUDY OF UPPER-LAYER CASES

	Link association	ICMP (Ping)	UDP (TFTP) in dirty env.		UDP (TFTP) in clean env.		TCP (Telnet)
			BMP file	MP3 file	BMP file	MP3 file	
ERR	100%	99.42%	98.08%	99.70%	99.89%	99.14%	58.11%
Test case succ. ratio	100%	86.99%	0% (13.3% Transmitted)	0% (0.96% Transmitted)	99.87%	82.23%	0% (12% Transmitted)

and transport-layer (UDP and TCP) protocols, and compare the results in terms of the ERRs of the link layer. Because the link-layer behaviors can be handled by EAR, the ERR is consistent with the success ratio of link association. For the ICMP, the ERR is slightly higher than the number of successful Ping messages because a portion of Ping messages fails in the real environment due to the corrupted Ping Responses from the DUT.

For the UDP experiment, STA sends a 300 KB BMP file and a 2 MB MP3 file by TFTP. The ERR is close to 100%; however, no file can be successfully sent. The reason is that a portion of packets in the real environment is not captured correctly. When EAR reproduces the TFTP packets with a gap of sequence number, the TFTP server requests the TFTP client to retransmit the lost packets and does not merge the following packets. Another experiment operates in a clean environment and all the packets are captured correctly. In this case, most of the reproduced TFTP transmissions are complete. Therefore, defects of capture highly affect the effectiveness of traffic replay.

For the TCP experiment, AP and STA establish a Telnet connection. The result shows the low ERR and no successful Telnet connection establishment. A TCP connection establishment requires the three-way handshake used to negotiate the ACK and sequence numbers. However, the reproduced ACK and sequence numbers are inconsistent with the new negotiation. Thus, the TCP connection cannot be established correctly so that the following packets are also rejected. As a result, the subsequent events fail because the Telnet server does not generate the subsequent packets appeared in the real environment. Upper-layer protocols can degrade the ERRs because actions of the DUT might be affected by upper-layer protocols.

V. CONCLUSION AND FUTURE WORKS

This work provides a real traffic replay solution with consideration of environment emulation to reproduce traffic behaviors in a real environment. The proposed EAR defines a series of events that follow the IEEE 802.11 protocol. EAR uses the three-level automata to achieve packet-replay control and synchronizes the environment effect with each packet captured in a real environment. To examine how EAR approximates the real traffic behaviors, a performance metric, called the event reproduction ratio (ERR), is proposed to quantify the performance of traffic replay.

We implement EAR to evaluate the influences of packet-replay control and environment emulation. The experimental results show that EAR can achieve the ERR of 95.9% when addressing the non-atomic events. On the other hand, the ERR is higher than 92.45% in the fading environments. Moreover, the results of the case study examine the relationships between the ERRs and the upper-layer protocols. EAR can achieve a high ERR under the network-layer and transport-layer traffic.

Further issues of real traffic replay deserve future works. We observed that incomplete packet capture may result in a high ERR but a low success ratio of upper-layer protocols. The traffic capture technology in wireless environments needs to be further investigated. Moreover, the ERR may be degraded due to inconsistent upper-layer information because actions of the DUT may be highly affected by upper-layer protocols.

REFERENCES

[1] T.-H. Cheng and Y.-D. Lin, "Low-storage capture and loss-recovery stateful replay of real flows," Univ. National Chiao Tung, HsinChu, Taiwan, Tech. Rep., Jun. 2009.

[2] C. Rolland, J. Ridoux, B. Baynat, and V. Borrel, "Using LiTGen, a realistic IP traffic model, to evaluate the impact of burstiness on performance," in *Proc. SIMUTools*, 2008.

[3] C. Phillips and S. Singh, "Techniques for simulation of realistic infrastructure wireless network traffic," *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, 2009.

[4] M. Wan, N. Yao, Y. Liu, and H. Zhang, "A fast information reproduction method for HTTP in WLAN," *IEEE WCNIS*, pp. 365 - 369, Jun. 2010.

[5] V. Lender and M. Martonos, "Repeatable and realistic experimentation in mobile wireless networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, Dec. 2009.

[6] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-based models of delivery and interference in static wireless networks," in *Proc. ACM SIGCOMM Computer Communication Review*, 2006.

[7] G. T. Nguyen, R. H. Katz, B. Noble, and M. Satyanarayanan, "A trace-based approach for modeling wireless channel behavior," in *Proc. Winter Simulation Conference*, 1996.

[8] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko, and K. Raatikainen, "Seawind: a wireless network emulator," in *Proc. GI/ITG Conference on Measuring, Modeling and Evaluation of Computer and Communication Systems*, 2001.

[9] P. De, A. Raniwala, S. Sharma, and T. Chiueh, "Mint: a miniaturized network testbed for mobile wireless research," in *Proc. IEEE INFOCOM*, Mar. 2005.

[10] G. Judd and P. Steenkiste, "A simple mechanism for capturing and replaying wireless channels," in *Proc. ACM SIGCOMM E-WIND Workshop*, 2005.

[11] IEEE 802.11 Working Group, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," Sep. 1999.

[12] Tcpdump public repository, http://www.tcpdump.org.