國立交通大學

資訊科學與工程研究所

碩 士 論 文

在無線區域網路中運用環境仿真的真實流量重播

Real Traffic Replay over WLAN with Environment Emulation

研 究 生：李佩璇

指導教授：林盈達　教授

中 華 民 國 一 百 年 六 月

在無線區域網路中運用環境仿真的真實流量重播
Real Traffic Replay over WLAN with Environment Emulation

研 究 生：李佩璇　　Student：Pei-Hsuan Li

指導教授：林盈達　　Advisor：Ying-Dar Lin

國 立 交 通 大 學
資 訊 科 學 與 工 程 研 究 所
碩 士 論 文

A Thesis
Submitted to Institute of Computer Science and Engineering
College of Computer Science
National Chiao Tung University
in partial Fulfillment of the Requirements
for the Degree of
Master
in

Computer Science

June 2011

Hsinchu, Taiwan

中華民國一百年六月

在無線區域網路中運用環境仿真的真實流量重播

學生：李佩璇　　　　　　　　　　　　　　指導教授：林盈達

國立交通大學資訊科學與工程研究所

摘要

真實流量重播的技術在有線網路上已經是一項成熟的技術，它擁有真實流量的高複雜度及重播可重複測試的特性，故廣泛用於測試網路產品的效能上。它主要的概念是透過精準控制流量撥出的先後順序來保持和 DUT (Device Under Test)之間的互動。但要將流量重播的技術應用在無線網路上遭遇到很大的挑戰；因為無線網路上的通訊媒介既開放又缺乏保護，所有位於同一或鄰近頻道的裝置同時分享媒介的結果常造成干擾或雜訊影響接收訊號的正確性，且訊號在開放式空間中傳遞也面臨能量衰減的問題·故只有重播流量並不足以正確重製流量真實環境中的行為，也就無法達到用相同的情境測試 DUT 的目的。因此本論文提出一套以事件驅動重播的方法 EDASR，在重播流量的同時也必須控制環境的影響來滿足重製的有效性·錄製流量時也同時收集環境的影響因素(訊號強度，雜訊強度，干擾)，然後在 testbed 上在環境重製的情形下重播流量. 結果顯示 reproduction ratio 分別在 DUT 相依的流量和不理想的無線環境下仍能保持 95.9%和 88.7%的 reproduction ratio。反之當沒有應用 EDASR 的情況下, reproduction ratio 最差只有 20.6%和 0%的 reproduction ratio。

關鍵字：流量重播、真實流量、無線網路、無線區域網路、環境仿真

Real Traffic Replay over WLAN with Environment Emulation

Student: Pei-Hsuan Lee Advisor: Dr. Ying-Dar Lin

Department of Computer Science
National Chiao Tung University

Abstract

Real traffic replay is one of the solutions to test network devices in labs. Using packet traces captured in the real environment could hold more details than modeling. However, packet replay control is required to manage the interactions with the DUT, and environment effects highly affect the packet exchange sequences, especially in wireless networks. Therefore, to apply traffic replay in wireless networks requires not only packet replay control but also the reproduction of environment effects. In this paper, we propose a method called event-driven automata-synchronized replay (EDASR) to address traffic replay. EDASR translates traffic traces into events following the IEEE802.11 protocol. The automaton of each event is applied to achieve the packet replay control with the environment effects. To evaluate traffic replay solutions, a performance metric, named reproduction ratio, is given to quantify how traffic replay approximates the traffic to the real environment. The software implementation of EDASR demonstrates that 95.9% and 88.7% reproduction ratios could be retained under DUT-dependent traffic and un-ideal wireless environment, respectively. Without EDASR, the worst cases of reproduction ratios are 20.6% and 0% respectively.

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

Contents

List of Figures

List of Tables

Chapter 1 Introduction

Traffic replay is one of the solutions to test network products applied to complicated using scenarios. Based on the sources of traffic, traffic replay can be classified into the *trace-based* and *model-based* technologies. Model-based traffic replay utilizes statistical methodologies to build models of traffic patterns or environment effects, and provides key parameters to configure the traffic patterns. However, complicated details might be neglected under inaccuracy modeling. On the other hand, trace-based traffic replay uses the traffic traces and environment information recorded in the real environment. Using trace-based traffic replay to generate traffic, also called real traffic replay, could hold most of the details in real environment.

Real traffic replay is highly applied in wired networks. Somebody [1-2] studied the stateful issue of real traffic replay. The packet exchanges with the DUT (Device Under Test) should be precisely controlled following the protocol. For example, the next request should not be transmitted before receiving the response for the current request. However, the packet replay control in wireless networks suffers further challenges due to variant wireless environments. Because a wireless medium is open and shared, transmissions may suffer packet errors caused by fading, noise, and interference. Packet exchange sequences could be disturbed by environment effects. Even though environment effects are reproduced, poor coordination between packet replay control and environment emulation may result in adding the environment effects to the other packet exchange sequences. Therefore, to apply real traffic replay in wireless networks requires not only packet replay control but also the reproduction of environment effects coordinately.

A number of studies investigated packet replay control and environment emulation issues. For each issue, the related studies are categorized into model-based and trace-based approaches. A summary of the previous works is shown in Table 1.

Table 1 Related works of traffic replay issues over wireless networks

Issue	Type	Main idea	Extension
packet (reproduced) replay Control	Model-based	Use matrices to compute network behaviors	Workload patterns[3]
		Multi-level of traffic entities	IP traffic model [4][5]
	Trace-based	Bases on captured traffic and configurations	Selected traffic [6]

		Traffic replay	Replay HTTP traffic[7]
environment emulation	Model-based	Get the deliver probability model	MAC and PHY model [8]
			Use the SNR value [9]
		Get the error probability model	Use trans. rate, data size, distance [10]
		Use the user's configuration	Model mobility[11][12]
	Trace-based	Use different test scenarios	Changes the simulated environments by time [13]
		Replay RSS value	By time stamp [14]
		Attenuating the radio signals	large-scale-testbed and node mobility[15]

In packet (reproduced) replay Control issues, S. Karpinski [3] specific the matrix model to generates wireless workload to reproduces the real wireless network traffic performance. Moreover, C. Rolland [4-5] provides LiTGen, which is the IP traffic model which made of several levels to characterize different traffic entity. Alternatively, the trace-based approaches focused on capturing traces in real environments, the trace-based approaches focused on capturing traces in real environments. C. Phillips [6] extracted the timestamp and size in bytes from recorded traffic trace, and then generated the customized traffic based on then. Ming Wan [7] first recorded HTTP conversations in WLAN, then reclassified and reunion those conversations; finally, used multi-thread to replay HTTP traffic fast.

In environment emulation issues, some model-based approaches rely on the real-world measurements; V. Lenders[8] uses the PHY and MAC models and Reis et al. [9] uses the SNR value to model the delivery probabilities. Alternatively, Giao T. Nguyen [10] focus on the tracing and modeling of wireless channel errors. Other model-based approaches rely on the user's configuration of bandwidth limitation, delay, packet loss and so on; P. Zheng [11-12] use the central-control to configure the parameters of mobility and topology;[13] changes bandwidth, error properties and delay properties to model changes of mobility. Alternatively, Trace-based approaches such like G. Judd [14] and P. De [15] designs the hardware to emulate wireless channels; Judd [14] replays the recorded RSS by timestamp on a FPGA-based testbed, and P. De [15] provides and large-scale testing through radio signal attenuators and node mobility by mobile robots.

To the best of our knowledge, few previous studies devote to real traffic replay considering both of packet replay control and environment emulation. In this paper, we proposes a method called *event-driven automata-synchronized replay* (*EDASR*). The roposed method defines events to describe traffic behaviors and applies automata to control reproduced packets and environment effects. According the events following the IEEE 802.11 protocol, this paper defines the *reproduction ratio* to quantify the approximate level of traffic replay from the real environment. We realize EDASR as a software solution on Linux-based systems and evaluate its performance by the reproduction ratio with experiments.

The rest of this paper is organized as follows. Chapter 2 introduces the background of IEEE 802.11 and environment factors. Chapter 3 describes the design issues and the problem statement. The design and implementation details of EDASR is shown in Chapter 4. The performance evaluation of EDASR is investigated in Chapter 5. Finally, we conclude this work in Chapter 6.

Chapter 2 Background

This section highlights two identified aspects that are challenging for real traffic replay over WLAN. When performing the accurate replay two conditions should always be considered. First is the traffic behavior; because the replay traffic regarded as valid network traffic by DUTs, it must send out the correct packets in the correct order and direction so that it follows the IEEE 802.11 of protocols. Second, in wireless networks the environment effects have a strong impact on the data communication, so the error rate depends on the current environment effects like fading, noise, and interference. It is necessary to reproduce these effects as accurately as possible. The following gives the concepts of these traffic protocols of IEEE 802.11 and wireless communication system in the physical layer.

A. IEEE 802.11

I. Connection state

IEEE 802.11 protocol[16-17] provides authentication, association, reassociation, disassociation, and deauthetication services to establish or finish the connection state between the source and the destination station. There are three type of connection state:

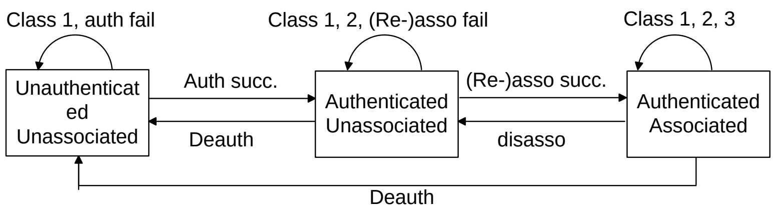
State 1: Initial start state, unauthenticated, unassociated.

State 2: Authenticated, not associated.

State 3: Authenticated and associated.

The current state existing between the source and destination station determines what kind of frame types can be transmitted between stations. If we want to replay the correct traffic behavior, we must keep the connection state is correct; otherwise those frames we sending out in the wrong connection state would be dropped by the DUT. Fig. 1 descripts the relationship and the transfer rule between connection states and the classes of frame types which are allowed to transmit in State1, State2 and State3. Through keeping the replay procedure at the correct connection state, we can reach the goal of sending out the correct packets in the correct order.

.



Class1	RTS, CTS, probe req., probe rsp., auth., deauth, broadcast data
Class2	Asso. req./rsp., reasso. req./rsp., disasso.
Class3	PS-Poll,deauth., any data

Fig. 1 Relationship between the connection states

II. Operations of frames

After knowing how to send out the correct packets by keeping the connection states at the requested statement, next key point is to ensure that atomic operations are not interrupted and the non-atomic operations can finish successfully. In wireless protocol, only sending out the correct packets in the correct order is not enough, it also sending out the correct packets with the right time interval due to the protocol.

Those atomic operations mostly need (1) short Inter-frame space and (2) positive acknowledgement to deal with, and other non-atomic operations (mostly are contention-based services) need a waiting time to deal with. For instance, the response time of the association response from the AP depends on the configuration of manufacturers or the communication link. The time requests are listed in the Table 2.

Table 2 Time intervals of different operations of frames

Time interval	Calculation	Frame type
SIFS		ACK, RTS, CTS, subsequence data and fragments.
DIFS	DIFS = SIFS + slot time + slot time	Before sending any data and management frames during a CP.
EIFS	Slot time + (8 x ACK Size) + Preamble Length + PLCP Header Length + SIFS + (2 x Slot time) Ps: ACK Size is 14 byte	Error occurred in previous receiving frame.
Response of the services	Depends on the configuration of manufacturers.	Contention-based services, ex. probe/auth/asso./reasso.(req.&rsp.) . or Deferred PS-Poll response. (AP may do regular DCF activities)

B. Wireless Communication System

Between the sender and receiver, the digital information across the wireless channel is by RF front-end. Fig. 2 shows a simple progress of transition and reception; the upper layer data frames or the mac layer control/management frame first passed through the encoder to eliminate unwanted regularity bits and add forward error correction (FEC). The modulator then converted bit streams into radio signal for electromagnetic wave transmitting. Finally, the transmitter added preamble and PHY header and transmitted radiated RF signal through antenna. The received side did an opposite process, and the principle is as sane as the transmitter.

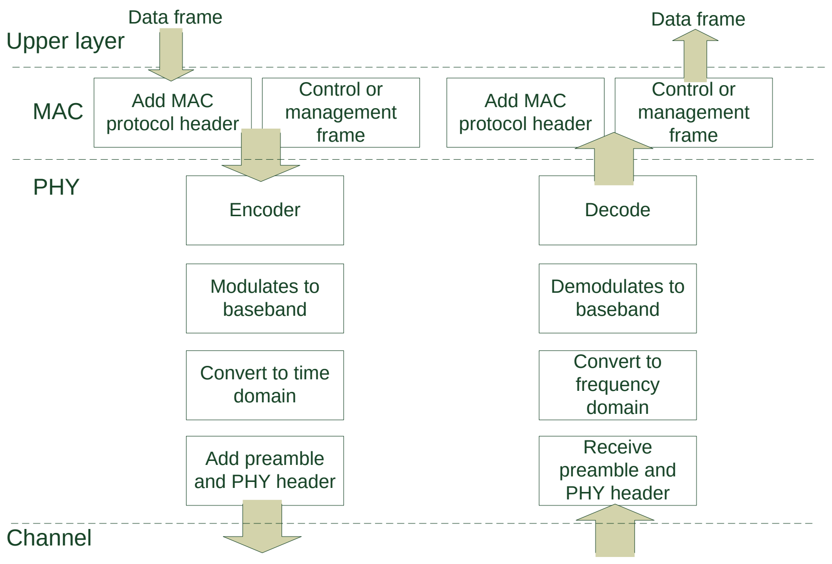


Fig. 2 The transition and reception over physical layer

During the propagation period on the channel, the signal which the receiver got at is affected by following effects: fading, interference and noise. There are two type of fading effects: large-scale and small-scale fading. Large-scale fading represents the path loss due to the decrease of signal strength by the distance. Small-scale fading refers to the changes in the amplitude and phase of signal, ex. multi-path fading and Doppler shift; Small-scale fading causes the signal is constructed or destructed at the receiver. In the same communication system, the interference is caused by the several sources transmit signal at the same time in the shared medium. The noise is a random disturbance signal in the different communication system, like microwave, Bluetooth and other devices which uses the same 2.4 GHz frequency channel.

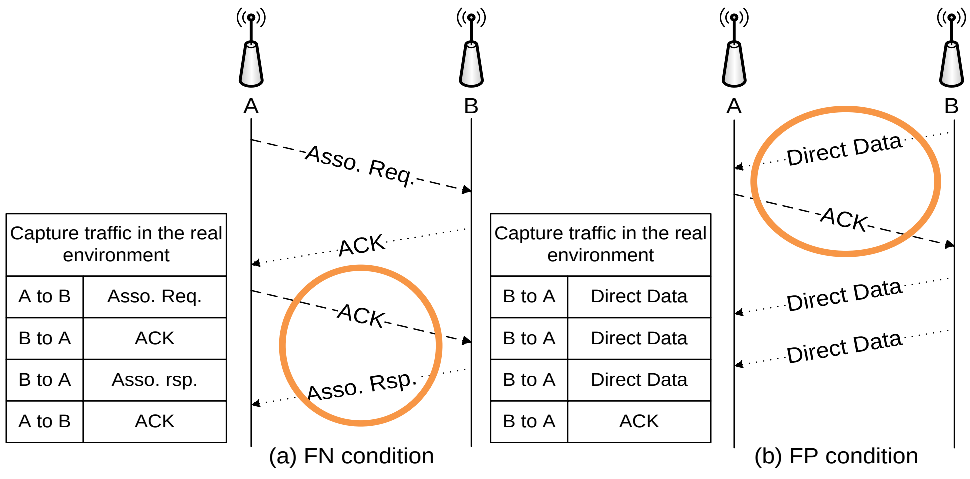
Fading, interference and noise all may cause the signal cannot be demodulated and decoded into originally correct bit stream and cause the increase of bit error rate, which affects the packet error rate, too. So if we want to reproduce the real traffic behavior, we also need to reproduce the environment effects as well. However, because the frequency, phase and amplitude are hard to capture and reproduce, in this thesis we mostly focused on using a software solution to emulate how strong the sender’s transmitted signal was got by the receiver and how strong the noise strength was got around the receiver, and the interference. Signal and noise strength need to control by the attenuation device or the signal generator; the interference can control by replay mechanism.

Chapter 3 Problem Statement

This chapter explains the issues of real traffic replay over WLAN and formulates the problem statements. Three types of causes may result in inconsistency of packet exchange sequences between the real environment and traffic replay. To analyze the traffic behaviors, EDASR defines events to transfer individual packets into events consisting of a sequence of packet exchanges. Based on the events, a performance metric called reproduction ratio are proposed to quantify the approximate level of traffic replay from the real environment.

A. Issues of real traffic replay over WLAN

Compared to the packet traces in the real environment, the reproduced traffic may suffer *false negative* (*FN*) and *false positive* (*FP*). In this paper, FN indicates a successful packet exchange sequences in the real environment fails in traffic replay. On the other hand, FP indicates a failed packet exchange sequences in the real environment succeeds in traffic replay. Loss of packet replay control may result in FN or FP cases. In Fig. 3a, the ACK from node A to node B is transmitted before the Association Response and results in invalidation of the association procedure which is successful in the real environment. In contrast, Fig. 3b shows a data frame is successfully transmitted following twice failures. However, the ACK does not wait for the corresponding data frame and lead the first data transmission succeeds due to no packet replay control.

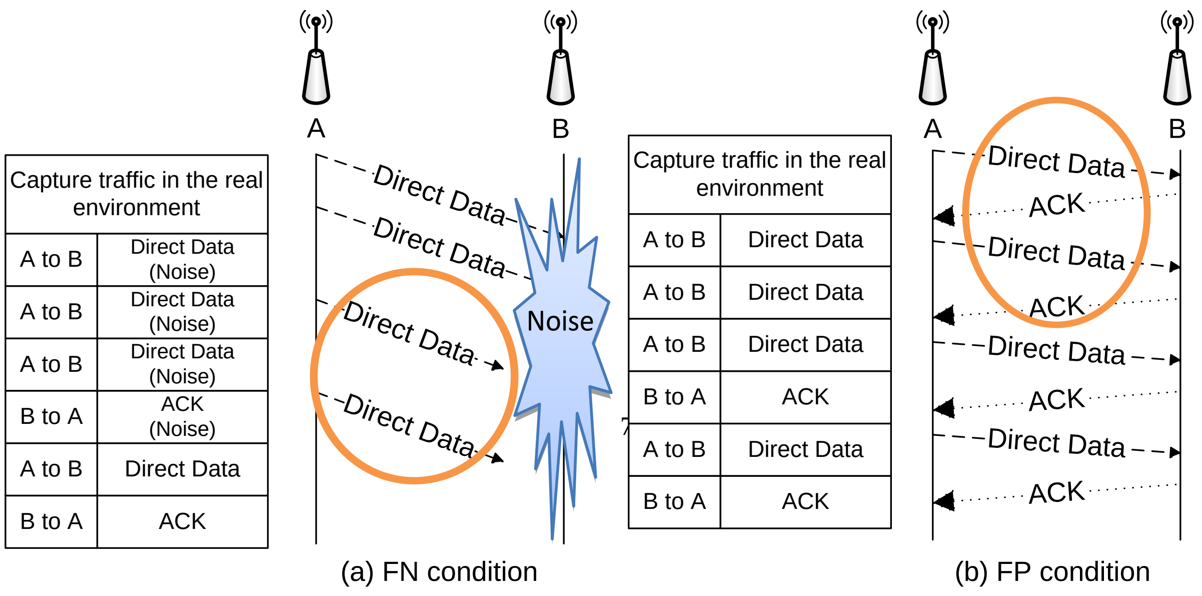


Capture traffic in the real environment	
A to B	Asso. Req.
B to A	ACK
B to A	Asso. rsp.
A to B	ACK

Capture traffic in the real environment	
B to A	Direct Data
B to A	Direct Data
B to A	Direct Data
B to A	ACK

Fig. 3 The lack of packet replay control

Transmissions in wireless networks are prone to bit errors due to variant environments. When the suffered bit errors exceed the tolerable threshold, a packet error occurs and might results in the failure of a series of packet exchanges, for example, the link association procedure. Thus, real traffic replay over WLAN requires both of packet replay control and environments. In Fig. 4a, reproducing the noise for an uncontrolled duration could lead the following two data frames to be involved in packet loss. Fig. 4b exhibits that transmission of data frames can become successful in traffic replay because the noise in the real environment does not be reproduced and disturb the packet exchanges.

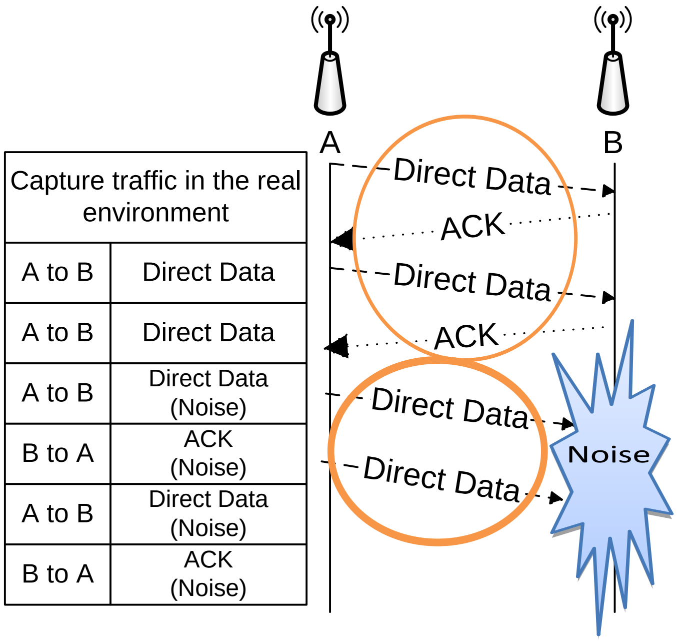


Capture traffic in the real environment	
A to B	Direct Data (Noise)
A to B	Direct Data (Noise)
A to B	Direct Data (Noise)
B to A	ACK (Noise)
A to B	Direct Data
B to A	ACK

Capture traffic in the real environment	
A to B	Direct Data
A to B	Direct Data
A to B	Direct Data
B to A	ACK
A to B	Direct Data
B to A	ACK

Fig. 4 The lack of environment effects

Even though the packet replay control and environment emulation is applied, the poor coordination between them can also cause failures. Fig. 5 shows that the noise happening to the wrong frames makes the frames failed to be received. However, the incorrect frames are received by node B. These examples illustrate lack of packet replay control or environment emulation may highly affect the packet exchanges sequences. Therefore, both of the packet replay control and environment emulation should be reproduced coordinately.



Capture traffic in the real environment	
A to B	Direct Data
A to B	Direct Data
A to B	Direct Data (Noise)
B to A	ACK (Noise)
A to B	Direct Data (Noise)
B to A	ACK (Noise)

Fig. 5 FN/FP due to poor coordination

B. Problem Formulation

To deal with the complex traffic behaviors, a formal statement to describe the packet exchange sequences is necessary. Because a single packet represents non-meaningful behaviors in the view of protocols, this paper defines *events* to represent meaningful packet exchange sequences according to protocol specification. For instance, a series of data and ACK frames following the RTS and CTS control frames could be treated as an event. Thus, a given packet trace could be transformed into a series of events. E_{Real} denotes the sequences of events captured in the real environment while E_{Replay} denotes the sequences of events reproduced by traffic replay. The *i*th events of E_{Real} is named $E_{Real}(i)$ whose value is 0 if the ith event succeeds. On the contrary, the value of $E_{Real}(i)$ is 1 if the *i*th event fails. Similarly, the value $E_{Replay}(i)$ can be 0 or 1 depending on the *i*th event of E_{Replay} is successful or failed.

By $E_{Real}(i)$ and $E_{Replay}(i)$, the problem statement could be given as follows. Given the captured packets and corresponding environment information, EDASR partitions the packets into a series of events, i.e. E_{Real}. According to E_{Real} and IEEE 802.11 protocol, EDASR reproduces the captured packets with the corresponding environment effects. The results of EDASR are also transformed into a series of events, i.e. E_{Replay}. To reproduce what happened in the real environment, EDASR intends to make E_{Replay} as consistent as possible with E_{Real} , In another words, $E_{Real}(i) = E_{Replay}(i), \forall i$. To quantify the approximate level of traffic replay, EDASR compares each event and uses the equation

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

where n is the number of events of E_{Real}, to compute the performance metric, called *reproduction ratio*, to examine the approximate level. Ideally, EDASR aims at maximizing the reproduction ratio. However, even though the environment effects is perfectly emulated, the probability that $E_{Real}(i)$ and $E_{Replay}(i)$ have an equal value is still not 1. Considering un-ideal wireless environments that results in packet error rates, $E_{Real}(i)$ could be 0 or 1 based on the packet error rates. Therefore, EDASR compares the average number of success to represent the value of each event. A parameter α is given as the threshold to determine whether $E_{Real}(i)$ and $E_{Replay}(i)$ is equal. That is,

$$\neg(E_{Replay}(i) \oplus E_{Real}(i)) = \begin{cases} 1, \text{if } \left|E_{Replay}(i) - E_{Real}(i)\right| < \alpha \\ 0, \text{otherwise} \end{cases} \tag{2}$$

Chapter 4 Event-Driven Automata-Synchronized Replay (EDASR)

This section describes proposed EDASR that is a traffic replay solution addressing the issues in Chapter 3. EDASR consists of three functions to achieve the real traffic replay. Transform into Events parses the captured traffic traces into a sequence of events, and then the event-driven automata-synchronized replay reproduces the events with environment effects by the automata. Finally, the evaluation function computes the reproduction ratios according to the events in the real environment and traffic replay. These functions are detailed in this section.

A. Overview

The captured traffic traces in the real environment exhibits only the sequences of packets included by traffic flows. However, each traffic flow acts according to the protocol. To reproduce traffic flows packet-by-packet represents no meaningful behaviors in the view of protocols. Thus, EDASR transforms packet sequences into events to address the traffic replay and evaluation. The flowchart of EDASR was presented in Fig. 6.

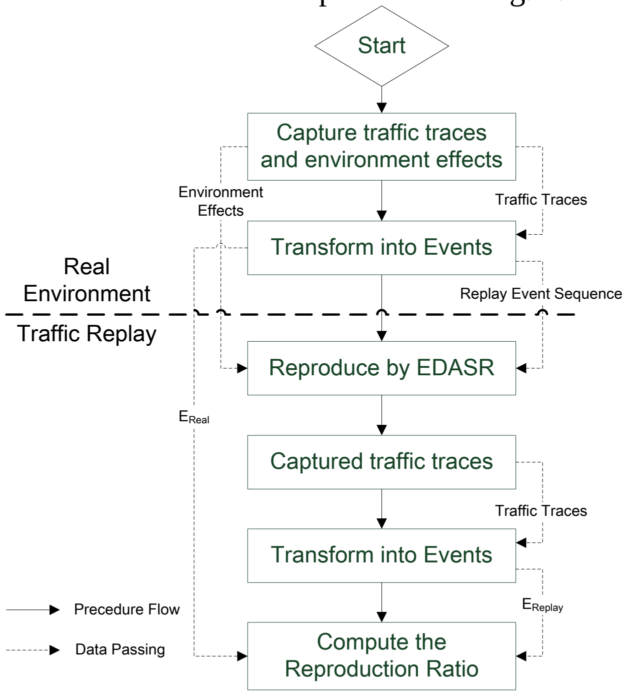


Fig. 6 Flowchart of EDASR

Because of the variance of wireless communication systems, the packets and environment effects seen by the transmitter and the receiver could be different. The environment effect of each packet through the wireless channel cannot be seen at the transmitter side whereas the traffic traces seen by the receiver could be corrupted. Therefore, the traffic traces and environment effects need to be observed at both sides. EDASR uses the traffic traces at the transmitter side, and the information of environment effects at the receiver side. Moreover, the captured traffic traces at the receiver side would be translated into E_{Real} for computing the reproduction ratio. On the other hand, the captured traffic traces at the other side would be translated into the replay event sequence. The replay event sequences and the environment effects at the DUT side are crossly combined for traffic replay.

Reproducing the replay event sequences should transmit each packet in correct orders and directions, and add the corresponding environment effects to the packet. EDASR defines the three-level automata to deal with various packet exchange sequences. EDASR transmits a packet with the environment effects after the packet exchanges with the DUT has been completed following the automata. On the other hand, EDASR adjusts the environment factors and then waits for the DUT returning a packet. Thus, the real environment can be emulated and synchronized with the packet exchange behaviors with the DUT. Finally, E_{Replay} could be obtained by capturing the traffic replay. E_{Replay} and E_{Real} would be compared to compute the reproduction ratio.

B. Transform traces into events

The captured traffic traces at the transmitter side represent the original traces which have not been corrupted by the environment effects, so there are suitable to replay. In contrast, the captured traffic traces at the receiver side represent the actual received conditions which been seen at the receiver, so we analysis the situations at the receiver to calculate the reproduction ratio. Fig. 7 shows the relationship of the traffic traces and environment factors between the real environment and replay.

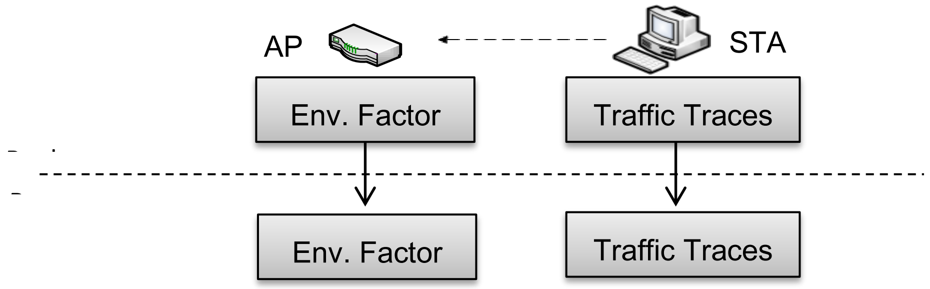


Fig. 7 The relationship of the traffic traces and the env. factor between the real environment and replay

Because the STA and the AP alternatively play the role of transmitter and receiver in a communication, we use the MAC address to determine that who is the transmitter and who is the receiver at each packet. After all, we need to transform both traces into replay event sequence for replay and E_{Real} for evaluating reproduction ratio. The events defined in this paper represent meaningful packet exchange sequences according to protocol specification; for keeping the correct interaction, we choose those *DUT-dependent* frames which affecting the behavior of DUT and then classify them into events, which were listed in Table 3.

Table 3 Component packets of events

Event type	Packets
Non-automic (AUTH,Asso,Reasso, Probe)	(Unicast MSDU Req,ACK), (Unicast MSDU Rsp, ACK)

Broadcast data	Broadcast data/ Beacon /Interference
Broadcast probe	(Broadcast probe, ACK)
Directed MMPDU	RTS, CTS, (data ACK)xn, (last data,ACK) (data ACK)xn, (last data,ACK) (last data,ACK)
Directed MSDU	RTS, CTS, (MSDU ACK)xn, (last MSDU,ACK) (MSDU ACK)xn, (last MSDU,ACK) (last MSDU,ACK)
Immediate PS-poll req.	(PS-poll , ACK)
Deferred PS-poll req.	(PS-poll , (data ACK)xn, last data ACK) (PS-poll ,last data, ACK)

Fig. 8 illustrates an example of how to transform traces into events. Monitor1 and Monitor 2 record the packets that are received and transmitted from the AP and the station(STA). First, Monitor2 recorded that the STA sent a data packet to the AP, and the packet was recorded by the Monitor1 means the AP surely got the packet. Second, Monitor1 recorded that the AP actually sent an ACK to response the STA, but the ACK did not occur in the recorded list in Monitor2 means that the STA missed the packet. Third, Monitor2 recorded that the STA sent second data packet to the AP and Monitor1 recorded that the AP got the packet. Finally, the ACK be seen by both monitors, therefore, the packet sent from the STA was received successfully by the AP. Next, we divided the recorded traces into two groups depending on the role of the receiver or transmitter of the communication, and then we Search Table 3 to cluster related packets into specific events. Events in the transmitter group are "replay event sequence" for replay, and events in the receiver group are "E_{Real}" for evaluating reproduction ratio.

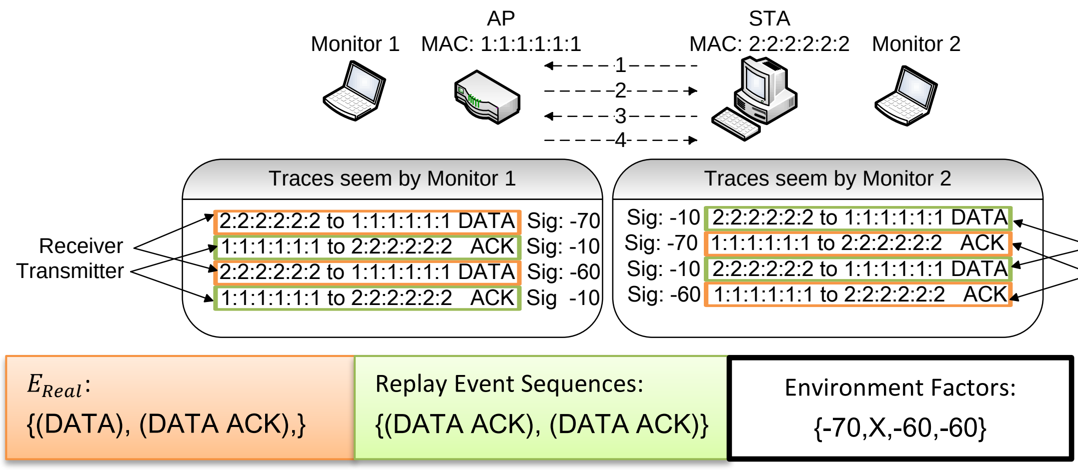

AP
MAC: 1:1:1:1:1:1
STA
MAC: 2:2:2:2:2:2
Monitor 1
Monitor 2
1
2
3
4
Traces seem by Monitor 1
2:2:2:2:2:2 to 1:1:1:1:1:1 DATA Sig: -70
1:1:1:1:1:1 to 2:2:2:2:2:2 ACK Sig: -10
2:2:2:2:2:2 to 1:1:1:1:1:1 DATA Sig: -60
1:1:1:1:1:1 to 2:2:2:2:2:2 ACK Sig -10
Traces seem by Monitor 2
Sig: -10 2:2:2:2:2:2 to 1:1:1:1:1:1 DATA
Sig: -70 1:1:1:1:1:1 to 2:2:2:2:2:2 ACK
Sig: -10 2:2:2:2:2:2 to 1:1:1:1:1:1 DATA
Sig: -60 1:1:1:1:1:1 to 2:2:2:2:2:2 ACK
Receiver
Transmitter
Transmitter
Receiver
E_{Real}:
{(DATA), (DATA ACK),}
Replay Event Sequences:
{(DATA ACK), (DATA ACK)}
Environment Factors:
{-70,X,-60,-60}

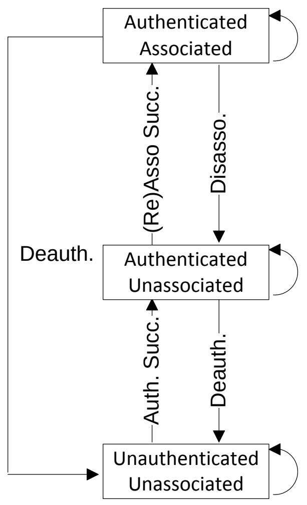
Fig. 8 An example of transforming traces into events

C. Replay by EDASR

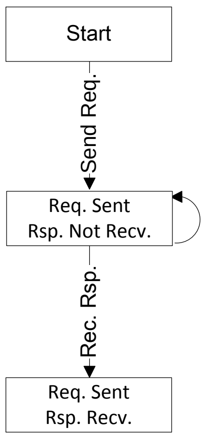
After gathering events from traffic traces, we further observe that there are two properties within events- *Non DUT-originated* and *DUT-originated.* The *Non DUT-originated* event means the first packet within the event do not be initiated by the DUT, so our replay machine has the power to decide when to send out packets. Whereas, the *DUT-originated* event means the event is initiated by the DUT so that DUT decide when to start transmitting packets and those events are mostly related to the management or control of connection states. Therefore, if the *DUT-originated event* unpredictably affects the replay process, we need to fix the effect or it would make the following replay events fail.

For keeping the correct interaction with the DUT, we design three levels of automata to handle the replay process, which are showed in Fig. 9 and Fig. 10. Level 0 is the highest level event, and this automaton maintains the connection state between the AP and station. If the *DUT-originated* events force the connection state transit to other state, we should to withdraw the effect. For instance, when the AP sent Deauth. or Disasso frames suddenly,. The Level 0 automaton should detect the change of the connection state and immediately take some measures to recover the connection state like sending authorization or association events frames to transfer the automaton state. The state transition of Level 0 automaton was Controlled by level 1 & level 2 automata.

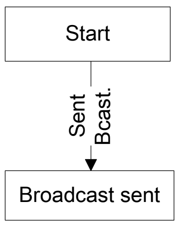
The can downgrade the current link state to the state which does not allow the data transition. For keeping the following data frame can be replayed successful, we need to send authorization and association request to make the link state transit to original state allowing data transitions. The state transition of Level 0 automaton was controlled by level 1 & level 2 automata; Fig. 9a presents the automata of level 0.



(a) The level 0 automaton



(b) The level 1 automaton

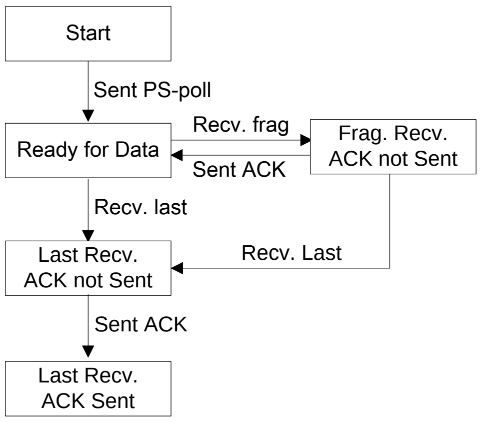


(c) Level 2 Broadcast

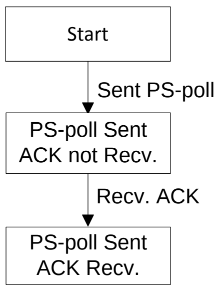
Fig. 9 The level 0, level 1 and level 2 automata

Level 1 automata deal with non-atomic events, which allow initialing another event before finishing the current event. For example, after sending the first packet (association request) in the association event, and before getting the association response from the destination AP, it is legal to start the probe event simultaneously. The state transition of level 1 automata are triggered by atomic events (level 2 automata) which will discuss in following paragraph and the automata of level 1 are presented in Fig. 9b.

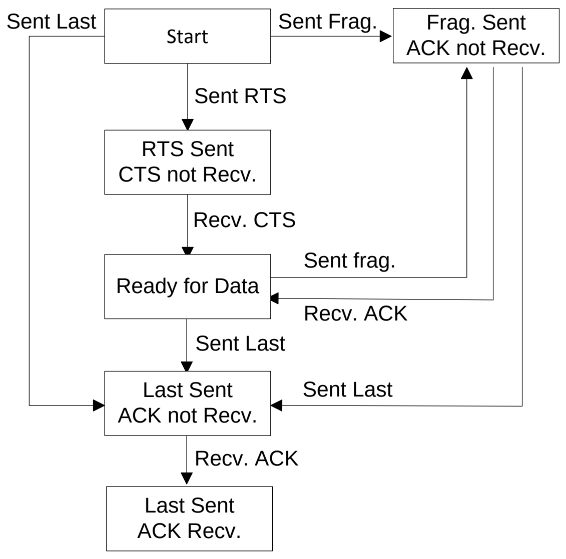
Level 2 automata that are presented in Fig.9c and Fig.10 are responsible for *atomic* operation sequences. The state transitions of the level 2 automata were triggered by the basic exchange of frames. Because the atomic operations in the atomic event must be performed entirely, we cannot insert other events before finishing the present event.



(a)Immediate PS-poll Rsp.



(b)Deffered PS-poll Rsp



(c) Directed MSDU/MMPDU

Fig. 10 The level 2 automata

Table 4 lists the component events within three levels of automata. One event may be involved in several levels, depending on which properties and functions it has.

Table 4 Event of three level automata

Automata Level	Automata	Event to Control
0	STA State	Maintain link 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

Using events as the replay units not only helps us to keep the correct traffic behavior conveniently, but also makes the synchronization between the traffic traces and the environment factors more easily. We have showed that if we do not control environment factors well, it will cause the FN and FP problems on replay. For keeping good coordination of both traffic traces and environment factors, we integrate the controlling component into level 2 automata which maintain the basic exchange of frames. Before transiting to the next state of the automata we will change the signal and noise strength to the target value. We show a simple control flow of synchronizing with environment factors in Fig. 11. At the initial state State0, we first set the signal value to -30dBm, noise value to -70dBm and then send the directed data frame out. After sending the directed data frame, the state was transited to State1, and now we are waiting for the ACK. In this moment we must prepare a suitable environment for the coming packet transmission, hence we need to change the signal value to -32dBm and the noise value to -70dBm.

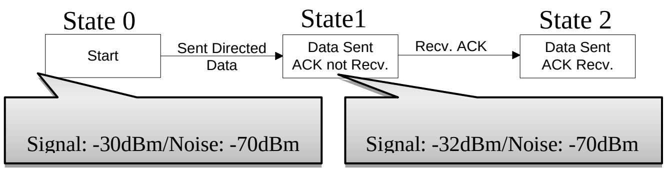


Fig. 11 Automata synchronize the environment factors and events

Chapter 5 Implement of EDASR

A. Overview

An installed EDASR host emulates one wireless device by two wireless network interface cards (WNICs), which are on behalf of the device's operating status of the transmission and the reception, and moreover, to emulate the path loss of signal strength by controlling the signal attenuator through GPIB. Fig. 12 illustrates the system components of EDASR. One WNIC for transmitting packets out is bases on the Linksys rt2870 chipset, and the other WNIC for listening the upcoming packets is bases on the Atheros AR5008 chipset. Using differ WNICs for the transmission and reception interfaces can improve the replay progress by separate the listen function into a thread. Secondly, the attenuator (EPA-1200B[20]) is used to adjudge the output signal strength of the transmission interface, and the time resolution is 20ms. Finally, the signal generator (Agilent E4438C[21]) is used to generate background AWGN (Additive White Gaussian Noise).

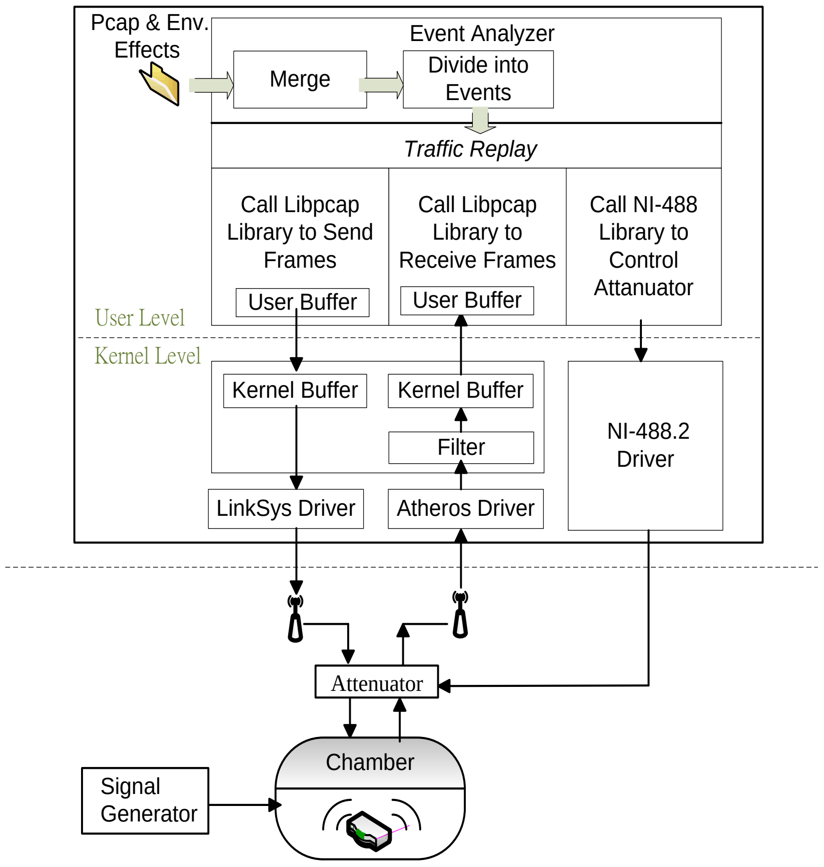


Fig. 12 Architecture of EDASR on the Linux

I. Event Analyzer

We use Omnipeek [18] as packet sniffer software because it supports the hardware and software solutions to capture information about wireless signal, noise and packets information. Each monitor which is installed Omnipeek can record the signal and noise value of each packet, moreover that total packets can be saved as a pcap file and the value of the signal and the noise can be saved as a text file. In the capturing phase in the real world, We arrange each AP or station with one monitor. Next, we use the method illustrated in the previous chapter to generate replay event sequence for replay and E_{Real} for evaluating reproduction ratio.

II. Traffic Replay

Since EDASR uses the functions of Libpcap [19] to send out and receive packets directly with the WNICs, we need to set our device into the "monitor mode". Monitor mode is a special driver setting which allow the WNIC to listen on the selected channel. Under the monitor mode, the captured packets are copied from the driver to the user program. Additionally, user can define the filter rule to determine which packets need to be pushed into upper layer. Libpcap supports the filter function through the BPF filter[18]. We can use pcap_compile() to compile the filter expression and pcap_setfilter() to load the filter into the packet capture device. Finally, each time when a packet was pushed into the user buffer, we can use the pcap_loop() or pcap_next() function to load the packet into our program.

Because most of the drivers does not support sending frame in the monitor mode, we need to modify the driver by insert the sending function in rt2870 driver. Fig. 13 shows the code flow of the Ralink rt2870 driver which we use for transmitting packets. We reference Aircrack-ng[22] to modify this driver for the packet injection function. First, rt28xx_packet_xmit() is the entry point between Linux kernel and driver, and STASendPackets() do early checking and classification for the outgoing packet. Second, RTMPDeQueuePacket() dequeues the outgoing frames from TxSwQueue. Third, STAHardTransmit() copies frames from waiting queue into the relative ring buffer, and then triger the hardware to send frames out to the air. Therefore, we insert the sending function MONITOR_Frame_Tx() is in STAHardTransmit() function.

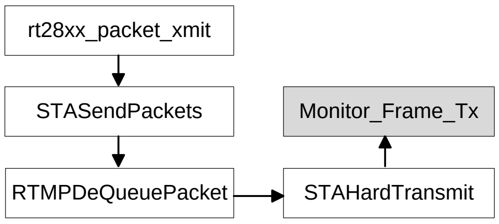


Fig. 13 Code flow for the rt2870 driver

Finally, The Traffic behavior in the replay process is controlled by those automata we defined in the previous chapter and simultaneously we change the signal strength by sending command to the attenuator through the NI-488.2 library [23].

Chapter 6 Evaluation

In this chapter, we use the reproduction ratio to investigate the performance of EDASR. The implementation is used to measure the improvement of EDASR and the effects of the traffic behavior and the environment effects are separately discussed.

A. *Evaluation Environment*

The experiment environment is shown in Fig. 14. The replay machine plays the role of a station connected to the AP in the real world, and monitors plays the role of sniffing the communication traffic in our replay period. The AP and the replay machine stay in different shield boxes and the attenuator lies between two shield boxes for emulating the path loss on the transmitted signal. Furthermore, the noise generator generates the different strength of the ambient noise around the AP, and the interference generator produces the interference packets in a specific time-frequency.

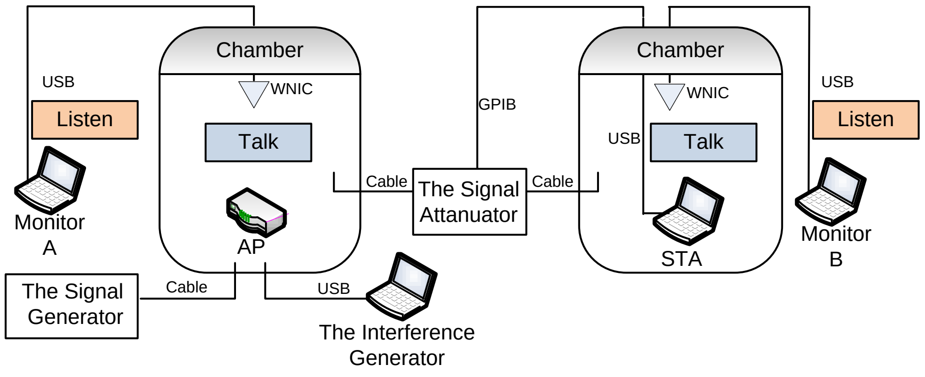


Fig. 14 Evaluation testbed

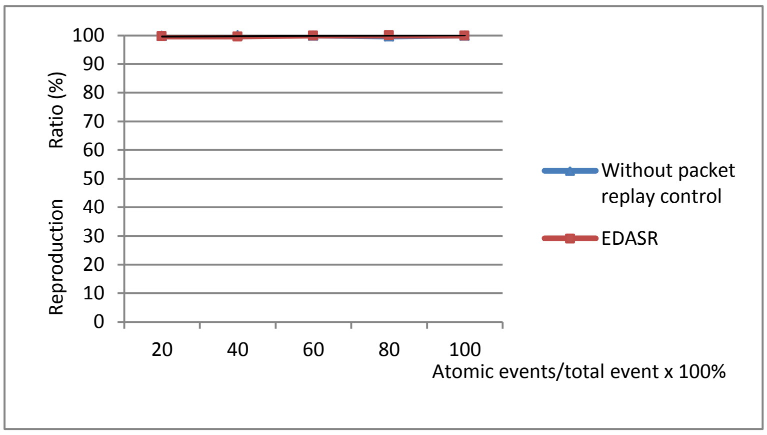
The parameters used in the evaluation of traffic control is about different proportion of atomic events and non-atomic events, and the parameters used in the evaluation of the environment effects are shown in Table 5. The parameters in Table 5 were collected in a simple real-world experiment. We use two notebooks installed with IXchariot [24] and one AP to generate the situation of transmitting IPTV traffic between the AP and the station. Furthermore, notebooks were located in different rooms (the distance is approximate 10m) to avoid face-to-face communication and enlarge the environment effects. Finally, we observe the strength of signal, noise and interference, and the frequency of interference at the same time by setting the monitors installed with Omnipeek [18] close to the AP and the station.

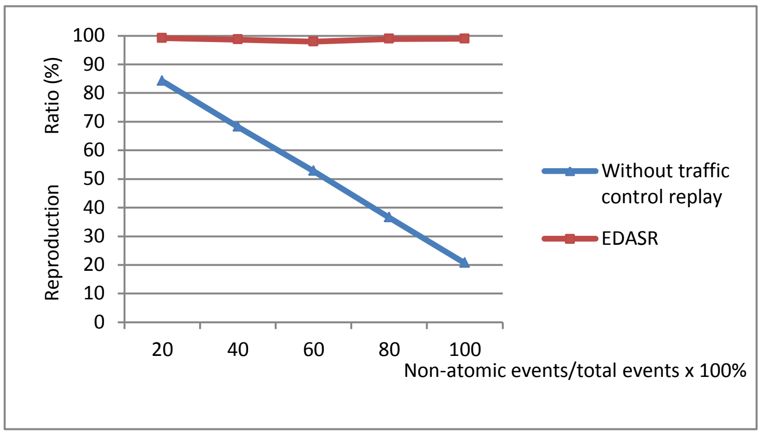
Table 5 Parameters of the environment effects

Parameter Name	Value
Traffic trace	IPTV traffic (total 7500000byte)
Signal strength	-64dBm
Noise strength	-85dBm
Interference period	40000μs
Interference strength	-69dBm

B. Reproduction Ratio Evaluation of Packet Replay Control

We defined the characteristics of atomic and non-atomic operations as atomic and non-atomic events in preview Chapter 4. In this experiment, we observe the reproduction ratio of EDASR in different proportion of events to examine how packet replay control affects the two kinds of events. We use simple unicast data event to present the atomic operation and use unicast probe event to present the request and response elements in non-atomic events. Fig. 15a shows the reproduction ratios of with EDASR and without EDASR are similar, since atomic events are timing-sensitive, the react time needed to follow the protocol (within the SIFS interval). Therefore, because the replay speed was not speed enough for interrupting the SIFS interval, the result could not feature that without packet replay control the data event would fail with error orders. Otherwise, in Fig. 15b, the result shows that EDASR obtained the 95.9% reproduction ratio in the worst case, and without packet replay control the reproduction ratio down to 20.68%. Because the non-atomic events require the request and response phase and the waiting time period is not fixed, EDASR supports the automata to control the waiting time period and make the packets arrived in the correct sequence.



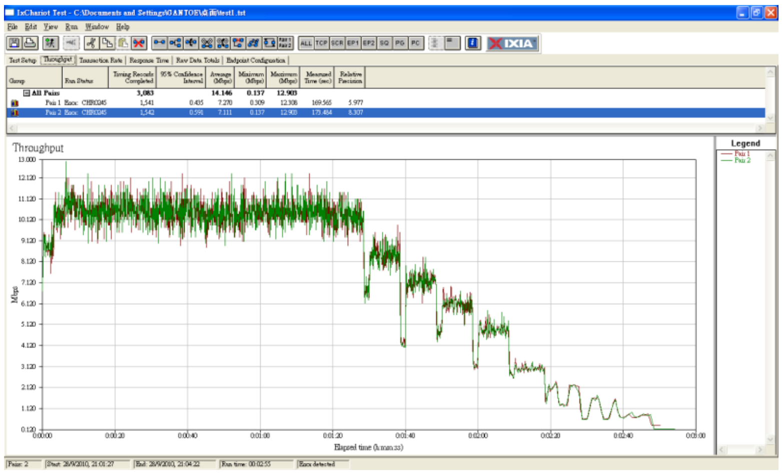


(a) Different proportion of atomic events (b) Different proportion of non-atomic events

Fig. 15 Reproduction ratio vs. proportion of atomic and non-atomic events

C. Reproduction Ratio Evaluation of environment Control

In chapter 4, we use software solutions and WLAN drivers to capture the information of fading, noise, and interference. Therefore, we can use that information to reproduce those environment effects. The section focuses on how the environment emulation affects the reproduction ratio. The results show the signal effects in Fig. 17a, the noise effects in Fig. 17b, and the interference effects in Fig. 18. First, we use a simple examine to observe the relationship between signal level and throughput; every one second the program control the attenuator to increase the attenuation value. The result showed looks like stairs in Fig. 16; the throughput down to zero along the increase of the attenuation value. The massive drop in SNR will affect the PER (Packet Error Rate) and cause the AP tries to resync at a lower speed. Finally, when the SNR value downs to something under the default threshold of the AP, the AP would not transmit any packets.

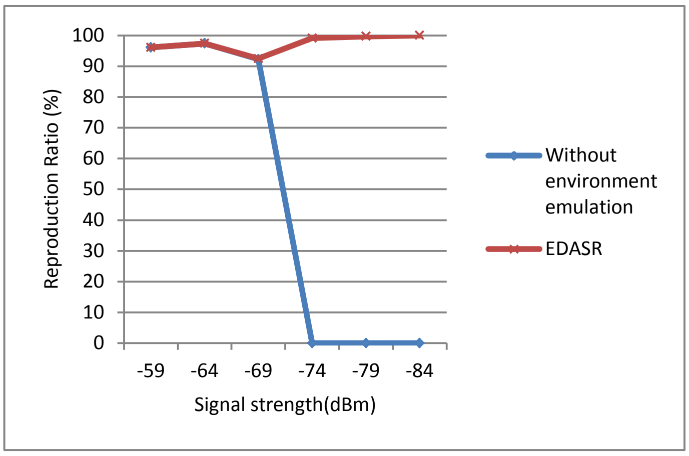


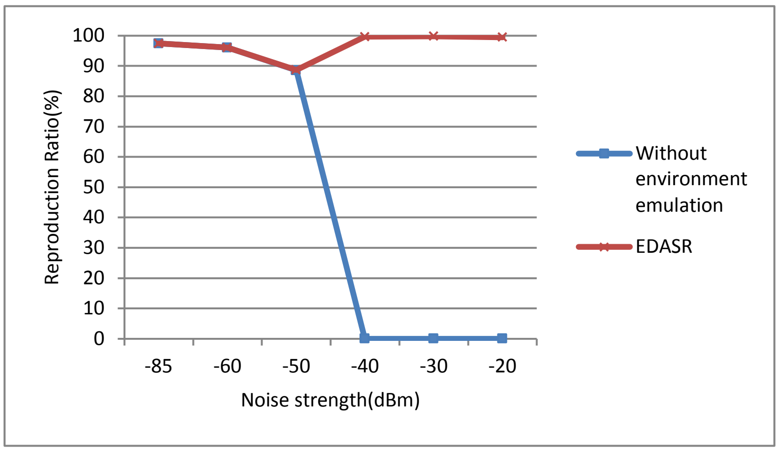
Group	Run Status	Timing Records Completed	95% Confidence Interval	Average (Mbps)	Minimum (Mbps)	Maximum (Mbps)	Measured Time (sec)	Relative Precision
All Pairs		**3,083**		**14.146**	**0.137**	**12.903**		
Pair 1	Exec: CHR0345	1,541	0.435	7.270	0.309	12.308	169.565	5.977
Pair 2	Exec: CHR0345	1,542	0.591	7.111	0.137	12.903	173.484	8.307

Pairs: 2 | Start: 28/9/2010, 21:01:27 | End: 28/9/2010, 21:04:22 | Run time: 00:02:55 | Error detected

Fig.16 The relationship between the signal strength and the throughput

Fig. 17a verifies the EDASR under different signal strength which range is between -59dBm and -85dBm. Similarly, Fig. 17b certifies the EDASR under different noise strength, which range is between -85dBm and -20dBm. Fig.17a and Fig17b can be discussed together; no matter changing the signal strength or the nose strength, the results are affecting the SNR. Fig.16a and Fig16b show that EDASR obtained the 88.7% reproduction ratio in the worst case and without the environment emulation the reproduction ratio down to 0%. Furthermore, observing those results we can discover that both curves in the graphs are concave at -69dBm and at -50dBm separately. The main reasons are: (1) under good environment conditions, the reproduction ratio is as good as the stable throughput which presents in the first who of Fig.16. Because of the change of SNR does not affect the PER heavily.(2) Under variation environment conditions, SNR has great extents of the PER which shown in the latter part of Fig. 16 (looks like stairs). Therefore, $E_{Replay}(i)$ has a probability to exactly match the $E_{Real}(i)$. (3) Under bad environment conditions, the results maintain the same concept which we illustrate in (1).





(a) Reproduction ratio vs. signal strength (b) Reproduction ratio vs. noise strength

Fig. 17 Reproduction ratio vs. signal and noise strength

Finally, Fig. 18 shows the result of the interference effects that implement EDASR or not seems similar and the reproduction ratio reaches 98.13% in the heaviest effect of interference. The reason is because that the collision avoidance mechanism of CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) and CCA (Clear Channel Assessment) deal with the wireless medium access through the physical carrier sensing. Consequently, if the devise senses that the channel is busy, it would wait until the channel is idle and then starts to transmit packets. Consequently, the reproduction ratio would not be affected heavily by interference. Theoretically, when the strength of interference is decreased under the CCA-threshold, the intolerable interference may raise the number of corrupted packets and cause the PER increased. As the results, EDASR would has more ability to reflect the relation of PER and the reproduction ratio.

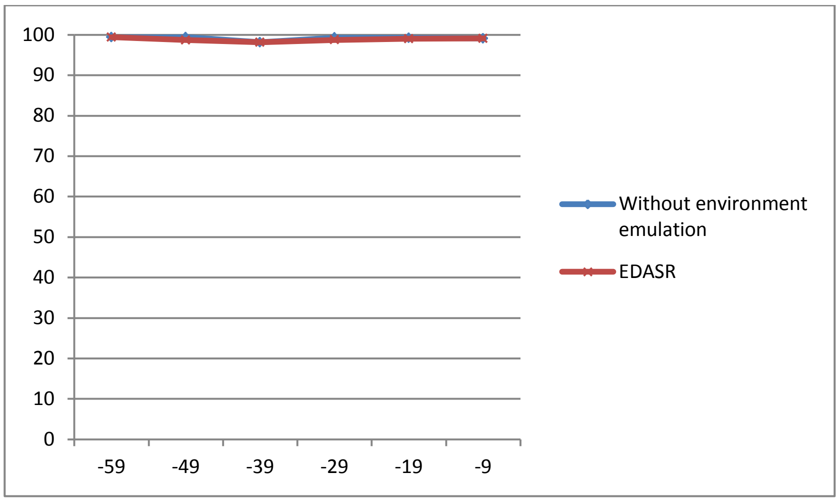


Fig. 18 Reproduction ratio vs. interference strength

Chapter 7 Conclusion and Future Works

Only replay traffic is not enough to make the traffic behavior as same as it in the real environment. Therefore, this thesis provides a solution to resolve the issues of replay of wireless traffic by emulating environment effectors simultaneously. The proposed method called Event-driven Automata-synchronized Replay (EDASR), which defines events to describe traffic behaviors and applies automata to control reproduced packets and environment effects. By analyzing the reproduction ratio we have proof that EDASR can provide the high accuracy of traffic replay from real environments. The software implementation of EDASR demonstrates that 95.9% and 88.7% reproduction ratios could be retained under DUT-dependent traffic and un-ideal wireless environment, respectively. Without EDASR, the worst cases of reproduction ratios are 20.6% and 0% respectively.

Our current work has proven that the methodology of EDASR through traffic control and environment emulation is useful. However, using application API to implement EDASR that would extend too much overhead in the context switching between the user space and the kernel space. In the current replay, the smallest transmission time between two packets is close to 1000μs, so some events fail due to the latency of packet. In the future work, we should extend EDASR to kernel modules or driver codes with some functionality of time-sensitive operations; consequently, we can overcome the drawbacks of performance and make EDASR support all time-sensitive operations (ex. ACK) and cover all event type of standard 802.11 protocol.

References

[1] Wu-Chang Feng, Ashvin Goel, Abdelmajid Bezzaz, and Wu-Chi Feng, “TCPivo: a high-performance packet replay engine,” Jonathan Walpole, in *Proceeding MoMeTools '03 Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research,* 2003.

[2] Tsung-Huan Cheng, and Ying-Dar Lin, “Low-Storage Capture and Loss-Recovery Stateful Replay of Real Flows,” *Univ. National Chiao Tung, HsinChu, Taiwan, Tech. Rep.,* Jun. 2009.

[3] Stefan Karpinski, Elizabeth M. Belding, Kevin C. Almeroth, and John R. Gilbert, “Linear Representation of Network Traffic,” *Journal Mobile Networks and Applications, Volume 14 Issue 4,* 2009.

[4] Chloé Rolland, Julien Ridoux, and Bruno Baynat, “LiTGen, a Lightweight Traffic Generator:Application to P2P and Mail Wireless Traffic,” in *Proceedings of PAM'07 Proceedings of the 8th international conference on Passive and active network*

measurement, Apr. 2007.

[5] Chloé Rolland, Julien Ridoux, Bruno Baynat, and Vincent Borrel, “ Using LiTGen, a realistic IP traffic model, to evaluate the impact of burstiness on performance,” in *Proceeding Simutools '08 Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008.

[6] Caleb Phillips, and Suresh Singh, “Techniques for Simulation of Realistic Infrastructure Wireless Network Traffic, ” *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, 2009. WiOPT 2009. 7th International Symposium on*, Jun. 2009.

[7] Ming Wan, Nan Yao, Ying Liu, and Hongke Zhang, “A fast information reproduction method for HTTP in WLAN,” *Wireless Communications, Networking and Information Security, 2010 IEEE International Conference on* , pp. 365 - 369 ,Jun. 2010.

[8] Vincent Lender, and Margaret Martonos, “Repeatable and Realistic Experimentation in Mobile Wireless Networks,” *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, Dec. 2009.

[9] Charles Reis, Ratul Mahajan, Maya Rodrig, David Wetherall, and John Zahorjan, “Measurement-Based Models of Delivery and Interference in Static Wireless Networks,” in *Proceeding SIGCOMM '06 Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, 2006.

[10] Giao T. Nguyen, Randy H. Katz, and Brian Noble, Mahadev Satyanarayanan, “A trace-based approach for modeling wireless channel behavior,” in *Proceeding WSC '96 Proceedings of the 28th conference on Winter simulation*, 1996.

[11] Pei Zheng, and Lionel M. Ni ,“EMWIN: Emulating a Mobile Wireless Network using a Wired Network”, *WOWMOM '02 Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, 2002.

[12] Pei Zheng, and Lionel M. Ni ,”EMPOWER: A Network Emulator for Wireline and Wireless Networks,” *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, Apr. 2003.

[13] Markku Kojo, Andrei Gurtov, Jukka Manner, Pasi Sarolahti, Timo Alanko, and Kimmo Raatikainen, “Seawind: A wireless network emulator,” in *Proceedings of 11th GI/ITG Conference on Measuring, Modeling and Evaluation of Computer and Communication Systems*, 2001.

[14] Glenn Judd, and Peter Steenkiste, “A simple mechanism for capturing and replaying wireless channels,” in *Proceedings of the 2005 ACM SIGCOMM workshop on Experimental approaches to wireless network design and analysis*, 2005.

[15] Pradipta De, hshish Raniwala, Srikant Sharma, and Tzi-cker Chiueh, “Mint: A miniaturized network testbed for mobile wireless research,” *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, Mar. 2005.

[16] P. Brenner, “A Technical Tutorial on the IEEE 802.11 Protocol,” BreezeCom, Jul. 18, 1996.

[17] Matthew S Gast, 802.11 Wireless Networks: The Definitive Guide, Second Edition, O'Reilly Media, Inc., 2005.

[18] OmniPeek Network Analyzer, http://www.wildpackets.com

[19] Libpcap, http://www.tcpdump.org

[20] E-INSTRUMENT TECHEPA-1200B, http://e-channel.com.tw

[21] Agilent Technologies, http://www.home.agilent.com

[22] Aircrack-ng , http://www.aircrack-ng.org

[23] NI-488.2 library, http://www.ni.com

[24] IxChariot, http://www.ixchariot.com/products/datasheets/ixchariot.html