L2Relay: Design and Implementation of a Layer 2 Wi-Fi Packet Relay Protocol

Shuaiyuan Zhou and Zhenghao Zhang
Computer Science Department
Florida State University, Tallahassee, FL 32306, USA
Email: {zhou,zzhang}@cs.fsu.edu

***Abstract*—In this paper, we propose L2Relay, a novel packet relay protocol for Wi-Fi networks that can improve the performance and extend the range of the network. A device running L2Relay is referred to as a relayer, which overhears the packet transmissions and retransmits a packet on behalf of the Access Point (AP) or the node if no ACK is overheard. One important feature of L2Relay is its ubiquitous compatibility, i.e., it is compatible with any Wi-Fi devices, such that one or multiple relayers can be installed in any network easily without any modification to the AP or the nodes. L2Relay is a layer 2 solution that exploits many layer 2 functionalities such as carrier sense. It encompasses unique solutions to link quality measurement, rate adaptation, and relayer selection. We implement L2Relay in the OpenFWWF platform and compare it against the baseline network without a relayer as well as a popular commercial Wi-Fi range extender. Our results show that L2Relay achieves overall better performance than both compared schemes.**

I. Introduction

Wi-Fi has been widely adopted in homes, offices, and public hot spots. One challenge that often arises is the range of the network, because Wi-Fi operates under strict transmission power limits, such that nodes on the edge of the network often get degraded or even interrupted service. This problem can be alleviated to a degree by range extenders such as [1], which extend the range of the network by capturing and rebroadcasting the packets. However, it is known that the range extenders may reduce network speed because the node may be close to the Access Point (AP) and can receive from the AP directly such that rebroadcasting the packets is unnecessary.

In this paper, we design a layer 2 packet relay protocol, called *L2Relay*, and implement it in the *OpenFWWF* open source firmware [15]. Our motivation is that a relay device should avoid relaying packets that have already been received; even for a lost packet, the relay device should determine whether or not to relay the packet based on channel conditions instead of always relaying it, because sometimes the original sender is in a better position to retransmit the packet. L2Relay is a layer 2 solution because it is independent of the physical layer while exploiting many layer 2 functionalities in wireless networks such as carrier sense. We refer to the relay device simply as *relayer* in the remaining of the paper.

We identify the following key requirements and challenges in the design of L2Relay:

- *Ubiquitous compatibility*: The relayer should be compatible with all existing types of APs and nodes requiring no modifications. Many problems can be significantly simplified if the AP or the node can be modified; however, the deployment of L2Relay will be much limited if L2Relay can only work with certain types of APs or nodes, or if device updates are needed which may be technically challenging for common users.
- *Link quality estimation*: The relayer should be able to measure the quality of the links to determine whether or not to relay for a node. For example, no relaying is needed if the link between the node and the AP is very good. The challenge is that the estimation should be robust and introduce minimum overhead.
- *Coping with multiple rates*: Wi-Fi supports multiple data rates with different communication ranges and loss ratios, which leads to many practical challenges. For example, the AP's rate selection algorithm may eventually adapt to the best rate to the relayer which may or may not be the best rate to the node. Also, the AP usually transmits at a subset of the rates such that the channel statistics at other rates may not be available, including the optimal rate.
- *Distributed relayer coordination*: As multiple relayers may be installed in a network, the relayers should be able to converge to the best relayer for any specific node. The challenge is that the relayers should reach an agreement without causing much overhead to the network while they may not be able to communicate with each other directly.
- *Low complexity*: The protocol should be simple and implementable in firmware because it needs to have access to the Medium Access Control (MAC) layer.

In this paper, we propose solutions to the above challenges. We test L2Relay with our implementation in the OpenFWWF platform in both emulated and real-world experiments. We compare L2Relay against the baseline network without a relay device as well as a popular commercial Wi-Fi range extender. Our results show that L2Relay achieves better performance than both compared schemes, which suggests that L2Relay is capable of adapting to the wireless channel conditions and taking appropriate actions.

The rest of the paper is organized as follows. Section II discusses the related works. Section III describes the L2Relay protocol. Section IV gives the experimental results. Section V concludes the paper.

978-1-4799-1270-4/13/$31.00

II. Related Work

Packet relay in Wi-Fi networks based on node cooperation has been studied extensively in recent years. The proposed protocols include PRO [3], Soft-Repeater [2], RCMAC [4], DAFMAC [5], rDCF [6], RAMA [7], and CoopMAC [8]. The fundamental difference between L2Relay and the existing protocols is that L2Relay is designed for ubiquitous compatibility without any modification to the nodes and the AP, while the existing protocols require such modifications. As a result, an L2Relay device can be easily installed in any Wi-Fi network and bring immediate benefits, while the existing protocols can only be installed in networks where a significant percentage of the nodes and the AP are modified. In addition, due to the different design requirements, L2Relay solves different and often more challenging problems than the existing protocols. For example, rate adaptation in L2Relay is very challenging because the unmodified sender will not differentiate between the ACKs from the relayer and the ACKs from the receiver. Furthermore, L2Relay is implemented in the firmware and field-tested, while existing protocols are either implemented in the driver and do not have access to much of the useful MAC layer information or are evaluated based on simulation.

Rate selection is a classic topic in wireless networks [10], [11], [12], [9]. L2Relay also handles rate selection by the sender and by the relayer. While existing rate selection algorithms can be adopted for the link between the relayer and the receiver, the rate selection between the sender and the relayer is different and challenging because the relayer has to select a rate for the sender when the sender is not even aware of the relayer. As a result, the solution in L2Relay is new and different from the existing algorithms.

Maranello [13] is also implemented on the OpenFWWF platform with a different focus on partial packet recovery instead of packet relaying.

III. L2Relay Design

We explain the L2Relay protocol in this section under a simple scenario with one AP and one node and with only the downlink traffic. We note that this is sufficient for describing the protocol because: 1) the operations on the downlink and the uplink are identical except reversing the roles of the AP and the nodes, 2) although there may be multiple nodes in the network, the operations to all node are identical, and 3) although a relayer may serve multiple APs if there are multiple APs in the network, the operations for each AP are identical. We note that the L2Relay relayer can improve the network performance in more complicated sceneries with more APs and more nodes, because for each packet transmission, if the relayer intervenes, it should usually reduce the air time consumption.

A. Notations

We in most cases use R to denote a relayer and N to denote the node. R may or may not relay for N depending on the channel conditions; if it does, we say R is *the relayer of* N or is *relaying for* N. As illustrated in Fig. 1, we denote the link from the AP to N as L_1, the link from the AP to R as L_2^R, and the link from R to N as L_3^R. The reverse links are denoted as L_1', $L_2'^R$, and $L_3'^R$, respectively. The Packet Receive Ratio (PRR) of a link is the fraction of packets that are received correctly, which is denoted as μ, along with the data rate, as PRR is related to the data rate. For example, the PRR of L_2^R at rate ρ is denoted as $\mu_2^R[\rho]$. The PRR notation may be simplified if the link and the rate are clear from the context. The *relay rank* of R is denoted as K_R and is calculated according to the PRRs where a better relayer has a smaller rank value.

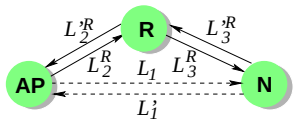

Fig. 1. Notations of links.

B. Overview of L2Relay

At initialization, R is manually supplied with the MAC address of the AP it should serve. Then, R monitors the channel and maintains PRR records for N; the PRRs for different rates are stored separately. Whenever possible, R uses the overheard data packets and ACKs to estimate the PRRs which is referred to as the *passive measurements*; it may also use *link probing* to gather data for rates with no passive measurements. R makes decisions based on the PRRs; for example, if $\mu_1[\rho]$ is much larger than $\mu_3^R[\rho]$ at some rate ρ, it should not relay for N. Suppose R is the relayer of N. After the AP sends a packet to N, R transmits the packet again if it does not detect the ACK from N. R runs a simple rate selection algorithm to adapt to the best rate for link L_3^R. R also periodically enters a *rate exploration mode* to force the AP to sample certain data rates in order to discover better data rates. To discover potential better relayers of N, R sends queries periodically to other relayers with a technique called *ACK reflection*. A better relayer Q may respond to the query and subsequently become the relayer of N.

C. Packet Relay Procedure

The basic packet relay procedure is described as follows.

1) High Level Description: Suppose R is the relayer of N. After the AP finishes sending a packet $\mathbf{P}$ to N, R will relay $\mathbf{P}$ if R received $\mathbf{P}$ correctly but did not detect the ACK from N for $\mathbf{P}$. If R decides to relay $\mathbf{P}$, it first sends an ACK to the AP on behalf of N such that the AP will not retransmit $\mathbf{P}$. R then transmits $\mathbf{P}$. After the transmission of $\mathbf{P}$, if R receives the ACK from N, it removes $\mathbf{P}$ from the buffer; otherwise, it will attempt to retransmit $\mathbf{P}$ for up to another $W-1$ times, where W is set to be 3 in our current implementation, after which it removes $\mathbf{P}$ from the buffer. Fig. 2 shows a typical packet transmission procedure captured by the GNU Software Defined Radio [18].

2) Details: We explain the details in the following.

Buffer management and packet transmission: For simplicity, R buffers only one packet. R writes an overheard packet to the buffer only if there is no packet in the buffer. The packet will be removed if an ACK from N has been received, or if the packet has been transmitted W times. When relaying a packet $\mathbf{P}$ for the first time, R transmits $\mathbf{P}$ SIFS (10 μs in 802.11g) after its ACK to the AP. If $\mathbf{P}$ is still not received by N, the retransmissions are subject to the same random backoff procedure in 802.11 DCF.

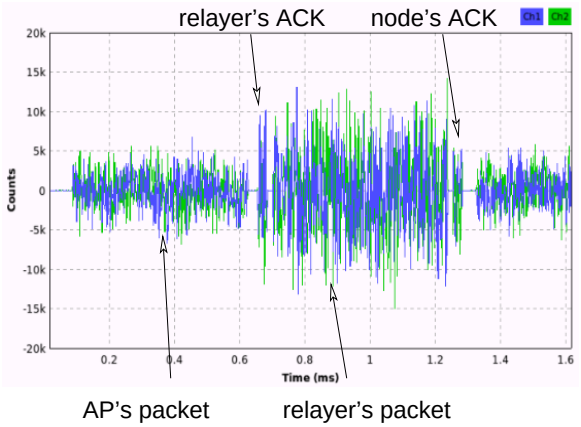

Fig. 2. A typical packet transmission procedure.

ACK detection: R needs to determine if an ACK has been sent by N to determine whether or not to relay $\mathbf{P}$. This may seem straightforward because R may simply check if it has received the ACK. However, this means that R will have to wait until the ACK transmission will have been completed. We note that this might have already been late because if N did not send the ACK, by this time, the AP may have already scheduled a retransmission, which basically defeats the purpose of using R for the retransmissions of lost packets. Therefore, L2Relay detects ACK by *checking whether or not the medium is busy when the ACK is supposed to be transmitted*, which can be carried out during the transmission of the ACK before the completion of the entire ACK. This is because the ACK should be sent SIFS after $\mathbf{P}$ while data packet transmission must wait for at least DIFS (28 μs in 802.11g) after $\mathbf{P}$; therefore, if the medium is busy shortly after $\mathbf{P}$, e.g., 15 μs after $\mathbf{P}$, it is due to the ACK with high probability. The exact amount of time to wait before checking the medium depends on the hardware platform. With our wireless card, we have to wait until at least part of the Physical Layer Convergence Procedure (PLCP) header of the ACK has been received, because our wireless card performs carrier sensing by receiving the PLCP header. We note that: 1) if R decides to be the relayer of N, it can detect the ACK reliably with very low false negative ratio, because it must be able to communicate with N at a non-trivial data rate while carrier sensing is much more robust than data communication, and 2) false positive should also be rare, most likely due to non-Wi-Fi devices or interferences from other Wi-Fi networks; even when false positive occurs, R will just miss a relay opportunity and may relay next time. We have carried out experiments which show that as long as R can at least communicate with N at the same rate N sends ACKs to the AP, the false negative and false positive ratios are very small, e.g., typically no more than 1-2%.

Relay frame modification: R relays every MAC frame byte, except replacing the source address in the MAC header with its own MAC address because it needs to receive ACK from N to determine if a retransmission is needed.

3) Loss Events: We now discuss the robustness of the relay procedure against packet loss.

We first consider when the AP sent a packet $\mathbf{P}$ and R received $\mathbf{P}$ correctly. There are two cases depending on whether N received $\mathbf{P}$ or not: 1) if not, N will not send ACK and R will relay the packet, which is the designed behavior of the protocol; 2) if yes, N will send ACK and if this ACK is detected by R, R will be silent according to the protocol, otherwise it will relay the packet, which is not necessary. We can see that the only problematic case is when both N and R received $\mathbf{P}$ correctly but R failed to detect the ACK from N. We note that *this is a rare event*, because as explained earlier, ACK should be detected reliably if R is a good relayer of N. In addition, we note that *the cost is bounded*. When R missed the ACK from N, it will first send ACK to the AP, then attempt to relay $\mathbf{P}$. The ACK from R will collide with the ACK from N, and the AP may receive neither of the ACKs. Therefore, the cost is no more than the AP's retransmission of $\mathbf{P}$ to N plus R's transmission of $\mathbf{P}$ to N. The cost is bounded because the AP and R have retransmission limits. Actually, the AP and R may get ACK from N and stop before reaching their retransmission limits, because their transmissions are unlikely to collide due to the random backoff mechanism.

We next consider when the AP sent a packet $\mathbf{P}$ and but R did not receive $\mathbf{P}$ correctly. We first note that this should not happen very often because R should be able to receive most packets from the AP if it can serve as a relayer of N. Not having received $\mathbf{P}$ correctly, R will be silent after the AP's transmission, but can get involved in the next packet transmission according to the protocol, either when the next transmission is a new packet or is a retransmission of $\mathbf{P}$.

Therefore, the protocol can function properly in the presence of packet loss. We also note that the relayer may introduce packet duplications. However, packet duplication may occur even without the relayer, e.g., due to the unnecessary retransmission of the AP, and the upper layers are able to handle the duplications.

D. Relayer Rate Selection

If R is the relayer of N, R should adapt to the best data rate to N. Any good algorithm may be used; currently, L2Relay uses the *SampleRate* algorithm [9]. Basically, R maintains a *current data rate*, denoted as $\rho_r^{*,R}$, which is initially set to the minimum rate. R measures $\mu_3^R[\rho_r^{*,R}]$ and $\mu_3'^R[\rho_r^{*,R}]$. A rate with smaller packet delivery time than $\rho_r^{*,R}$ under ideal PRR is called a *candidate rate*. Every 10 packets, R randomly selects a candidate rate as the rate for the first transmission of the packet to learn the PRR at this rate; the possible retransmissions are at $\rho_r^{*,R}$. $\rho_r^{*,R}$ is updated if a candidate rate is better.

E. Link Quality Estimation

One key aspect of L2Relay is the estimation of link PRRs. R runs a link quality estimation module and keeps the estimated PRRs in a table which can be queried by other modules. Separate estimate is maintained for each data rate. The estimation is based on the assumption that R can detect the ACKs from N reliably; at the end of Section III-E, we explain why this assumption will not lead to incorrect behaviors of the protocol.

1) Estimation with Passive Observations: R can passively observe the link and obtain estimations without sending any packet. In the following, we describe the estimations for one generic data rate ρ because the estimation method is the same for all data rates. For simplicity, we drop the rate and superscript in the notation of PRR. R can estimate μ_1, μ_1',

μ_2, and μ'_3 with passive observations; if it has no additional information, it assumes $\mu'_2 = \mu_2$ and $\mu_3 = \mu'_3$.

Basically, R maintains a set of counters to track the frequencies of certain events. When the AP transmits a fresh packet $\mathbf{P}$, the counters for the rate of $\mathbf{P}$ may be updated. The counters are:

- $C_{\mathbf{hP}}$: incremented by 1 whenever R receives the PLCP header of $\mathbf{P}$ correctly and the MAC header of $\mathbf{P}$ indicates the packet is from the AP.
- $C_{\mathbf{P}}$: incremented by 1 whenever R receives $\mathbf{P}$ correctly.
- $C_{\mathbf{PA}}$: incremented by 1 whenever R receives $\mathbf{P}$ correctly and detects the subsequent ACK from N.
- $C_{\mathbf{PAP}}$: incremented by 1 whenever R receives $\mathbf{P}$ correctly and detects the subsequent ACK from N, then receives $\mathbf{P}$ again with the same sequence number. The second transmission need not be at the same data rate as the first.
- $C_{\mathbf{PA}\wedge\mathbf{A}}$: incremented by 1 whenever R receives $\mathbf{P}$ correctly and detects the subsequent ACK from N, but fails to receive the ACK correctly.

Every 1 second, R calculates the PRRs according to the values of the counters, which are regarded as new samples of the PRRs. The PRRs are then smoothed according to the standard Exponential Weighted Moving Average (EWMA) with a coefficient of 0.5 for the new sample. If a counter serving as the denominator in an equation is too small, the sample is not valid and the PRR is not updated. If no sample is valid in the last 10 seconds, the PRR value is set to be invalid. After the update, the counters are cleared.

R estimates μ_1 by

$$\mu_1 = \frac{C_{\mathbf{PA}}}{C_{\mathbf{P}}} \tag{1}$$

based on the definition of μ_1. We note that this estimation is accurate if R can detect the ACK from N reliably.

R estimates μ'_1 by

$$\mu'_1 = 1 - \frac{C_{\mathbf{PAP}}}{C_{\mathbf{PA}}}, \tag{2}$$

because $C_{\mathbf{PAP}}$ is the number of events when the AP missed the ACK given R has observed the ACK.

R estimates μ_2 by

$$\mu_2 = \frac{C_{\mathbf{P}}}{C_{\mathbf{hP}}}. \tag{3}$$

We note that this would be an accurate estimate of μ_2 if R can detect all packets from the AP and include them in $C_{\mathbf{hP}}$. However, there are packets with corrupted PLCP headers and MAC headers which will not be counted in $C_{\mathbf{hP}}$; therefore, μ_2 may be overestimated. We still use this estimation method because the amount of overestimation is likely to be small for operative links.

R estimates μ'_3 by

$$\mu'_3 = 1 - \frac{C_{\mathbf{PA}\wedge\mathbf{A}}}{C_{\mathbf{PA}}}. \tag{4}$$

We note that it is easier to detect the ACK than to receive the ACK. The above equation determines μ'_3 based on the probability of not receiving the ACK correctly given an ACK has been detected.

One important feature of the passive estimation is that the equations above are valid even when another relayer, denoted as Q, is relaying for N. This is because the estimation of μ_1 and μ'_3 depend only on the packet transmission from the AP and the subsequent ACK from N before any possible involvement from Q. We note that R will not misinterpret an ACK from Q as an ACK from N based on the timing; the former is sent with some delay after SIFS while the latter is sent immediately after SIFS. The estimation of μ'_1 depends on $C_{\mathbf{PAP}}$; still, Q should be able to detect the ACK from N and will not relay this packet.

2) Probing: R may also send probes to learn the qualities of L_2^R and L_3^R, because the passive observations can be insufficient in certain cases. For example, if $\mu_1[\rho]$ is very small, R will not be able to estimate $\mu_3'^R[\rho]$ according to Eq. 4 because few ACKs from N can be observed at rate ρ. R has to rely on the passive observations to estimate the quality of L_1; on the other hand, it can be more actively involved in estimating the links it shares with N and the AP, because correct choices of rates are critical to the network performance. As the probing process is the same to N and to the AP, in the following, we explain only the probing to N.

For simplicity, the purpose of the probing is to discover the optimal data rate for L_3^R, defined as the highest rate such that the PRR is above a threshold β, which is set as 0.8 in our current implementation. To limit the overhead, R should send probes infrequently, which is one beacon interval every 100 beacon intervals in our current implementation. During the probing beacon interval, R sends η dummy packets at each selected rate to N, where η is 20 in our current implementation. The dummy packets are of minimum size and are not retransmitted if no ACK is received. In our current implementation, the optimal rate is found by a binary search based on the assumption that the PRR is a non-increasing function of the data rate. To be more specific, the search starts by probing a rate in the middle. If its PRR is above β, all rates below the probed rate are marked off; otherwise, all rates above. The search then resumes on the remaining rates, until both PRRs above β and PRRs below β have been found. The passive observations can also be used to reduce the search space. That is, if the passive observations indicate low PRR at a certain rate, all higher rates can be marked off; similarly, if the passive observations indicate high PRR at a certain rate, all lower rates can be marked off.

R uses the following counters for each rate where we again drop the superscript and the rate in the notation for simplicity:

- $C_{\mathbf{rP}}$: incremented by 1 whenever R sends a dummy packet.
- $C_{\mathbf{rPA}}$: incremented by 1 whenever R sends a dummy packet and detects the subsequent ACK from N.
- $C_{\mathbf{rPA}\wedge\mathbf{A}}$: incremented by 1 whenever R sends a dummy packet and detects the subsequent ACK from N, but fails to receive the ACK correctly.

R estimates μ_3 by

$$\mu_3 = \frac{C_{\mathbf{rPA}}}{C_{\mathbf{rP}}}. \quad (5)$$

R estimates μ_3' by

$$\mu_3' = 1 - \frac{C_{\mathbf{rPA} \wedge \mathbf{A}}}{C_{\mathbf{rPA}}}. \quad (6)$$

When R receives the next beacon packet from the AP, it calculates the PRR values for the rates with measurements collected from link probing, and replaces the old PRR values of such rates, after which it clears the counters. This is because the PRRs calculated with link probing are mostly for the rates with no passive observations and the old PRRs are most likely collected at least 100 beacon intervals earlier.

3) Estimation when Relaying for N: When R is actively relaying for N, it can get more frequent observations. For simplicity, we describe the operations for one generic data rate ρ and drop the rate and the superscript in the notation. R can estimate μ_3 and μ_3' in a similar manner as in link probing. In addition, it can get more accurate estimate for μ_2'. It maintains counters:

- $C_{\mathbf{rA}}$: incremented by 1 whenever R sends an ACK.
- $C_{\mathbf{rAPrx}}$: incremented by 1 whenever R receives a retransmitted packet from the AP after sending an ACK.

It calculates

$$\mu_2' = 1 - \frac{C_{\mathbf{rAPrx}}}{C_{\mathbf{rA}}}. \quad (7)$$

R updates μ_2', μ_3, and μ_3' every second according to EWMA with a coefficient of 0.5 for the new sample. The counters are then cleared. No update is made if the new sample is not valid. A PRR is cleared if no sample is received in 10 seconds.

4) If R Cannot Detect the ACK from N: As mentioned earlier, the estimation of PRR is based on the assumption that R can detect the ACKs from N reliably. If this is not true, certain PRR values will be incorrect. For example, $C_{\mathbf{PA}}$ will be smaller than the actual number of packets received by N such that μ_1 will be underestimated according to Eq. 1. Fortunately, in this case, the channel between R and N must be very poor such that the estimated μ_3 and μ_3' for any rate will be either invalid or very small. As a result, R will never consider itself a good relayer of N and the incorrect estimations of PRRs will not matter.

F. Expected Delivery Time Calculation

If the PRRs of all links are given, R can calculate T_{ρ_a,ρ_r}, which is the expected delivery time of a packet $\mathbf{P}$ of unit size when the AP and R transmit at rate ρ_a and ρ_r, respectively. T_{ρ_a,ρ_r} can be found based on a recursive relation. As we have to implement the calculation module in firmware, we limit the complexity of the module by making the following simplifying assumptions:

- If R receives $\mathbf{P}$ from the AP and sends an ACK, this ACK will be received by the AP. We note that in practice, given that R just received $\mathbf{P}$ from the AP correctly, the channel is likely in a good condition such that the ACK will likely be received by the AP.
- The receptions of other packets and ACKs are independent.
- R can always detect the ACK from N.
- The AP and R will keep retransmitting $\mathbf{P}$ until an ACK is received.

T_{ρ_a,ρ_r} can be derived according to a case analysis:

- *Case 1*: If neither N nor R receives $\mathbf{P}$, the process is repeated.
- *Case 2*: If N receives $\mathbf{P}$, N will send ACK. Regardless of whether or not R receives $\mathbf{P}$ correctly, if R can always detect the ACK from N, it will not attempt to relay $\mathbf{P}$. Depending on if the ACK is received by the AP:
  - If yes, one transmission is needed.
  - If not, the AP will retransmit $\mathbf{P}$ and the process is repeated.
- *Case 3*: If $\mathbf{P}$ is not received by N but by R, R will send the ACK and transmit $\mathbf{P}$. According to our assumption, the ACK from R will be received by the AP and one transmission is needed by the AP and $\frac{1}{\mu_3[\rho_r]\mu_3'[\rho_r]}$ transmissions are needed in expectation by R.

Therefore,

$$\begin{aligned} T_{\rho_a,\rho_r} &\approx (1-\mu_1[\rho_a])(1-\mu_2[\rho_a])(\frac{1}{\rho_a} + T_{\rho_a,\rho_r}) \\ &+ \frac{\mu_1[\rho_a]\mu_1'[\rho_a]}{\rho_a} + \mu_1[\rho_a](1-\mu_1'[\rho_a])(\frac{1}{\rho_a} + T_{\rho_a,\rho_r}) \\ &+ (1-\mu_1[\rho_a])\mu_2[\rho_a](\frac{1}{\rho_a} + \frac{1}{\mu_3[\rho_r]\mu_3'[\rho_r]\rho_r}) \end{aligned}$$

which can be simplified into Eq. 8:

$$T_{\rho_a,\rho_r} \approx \frac{\frac{1}{\rho_a} + \frac{(1-\mu_1[\rho_a])\mu_2[\rho_a]}{\mu_3[\rho_r]\mu_3'[\rho_r]\rho_r}}{\mu_2[\rho_a] + \mu_1[\rho_a]\mu_1'[\rho_a] - \mu_1[\rho_a]\mu_2[\rho_a]}. \quad (8)$$

G. Relayer Rank Calculation

R should know if it is a good relayer for N. The *rank* of R is denoted as K_R and is basically the minimum expected air time used such that the AP can send a unit size packet to N with the help of R, where the minimization is taken over the possible data rates. R calculates its rank based on Eq. 8 by plugging in different rates. A simplification can be made because R should always use $\rho_r^{*,R}$, the best rate to N, as ρ_r in Eq. 8. Therefore, the search is only over the possible rates the AP may use. R will not consider itself a candidate relayer if its rank is larger than the minimum packet delivery time for the AP without the help of R.

H. Rate Selection for the AP

Introducing a relayer to the network can affect the rate selection algorithm by the AP. For example, if R relays every packet of the AP, the AP will see a very good link and will

continue to increase the data rate. As a result, N may not be able to receive any packet from the AP directly, such that the link degrades into a two-hop path. The two-hop path may or may not be better than a single hop link that works partially. After much thought, we believe that if R acts as the relayer, it has to make efforts to lead the AP to the better rates, because rate selection is critical to the network performance and only R has the full knowledge about the links.

1) Rate Exploration: Let $\hat{\rho}_a^{*,R}$ be the best rate of the AP known to R, which is calculated by R based on its PRR tables according to Eq. 8. R has two modes, namely the *operating mode* and the *exploration mode*. It is in the operating mode by default but will also periodically enter the exploration mode, set to be 1 second in every 10 seconds in our current implementation. The packet relay policies are:

- Operating mode: Only relay packets at rates no higher than $\hat{\rho}_a^{*,R}$.
- Exploration mode: Only relay *retransmitted* packets at rates no higher than $\hat{\rho}_a^{*,R}\mu_2^R[\hat{\rho}_a^{*,R}]\mu_2'^R[\hat{\rho}_a^{*,R}]$.

We note that R does not relay packets at rates higher than $\hat{\rho}_a^{*,R}$ in the operating mode because this will lead the AP to selecting $\hat{\rho}_a^{*,R}$. The exploration mode is introduced to help R discover $\rho_a^{*,R}$, the actual optimal rate. We note that if the PRRs at all rates for all links are available, $\rho_a^{*,R}$ must be $\hat{\rho}_a^{*,R}$. However, R may only have measurements for a subset of the rates, especially for the link from the AP to N, because the AP will most likely transmit at only a subset of the rates, while this subset may not include $\rho_a^{*,R}$. Clearly, the key is to trigger the AP to probe as many rates as possible to get samples at more rates. We note that a rate selection algorithm will typically sample rates higher than its primary data rate; therefore, the challenge is mainly how to get samples at lower data rates. In the exploration mode, as only retransmitted packets are relayed, the AP may see significant losses at all rates higher than the rate at which it can directly communicate with N; therefore, it may drop the rate and visit many data rates. This policy is simple to implement because R may just check the retransmission flag in the 802.11 header and the rate to determine whether or not to relay a packet. It will not lead to a disastrous performance drop because the retransmitted packets are still relayed.

2) Analysis: We give an approximate analysis on AP rate selection. We define the performance ratio with respect to AP rate selection as the largest possible ratio of the calculated rank of R based on its available PRR values over the actual rank of R when the PRRs of all rates are available. Clearly, the ideal performance ratio is 1; however, it may be larger than 1 because R has knowledge only about a subset of rates. To avoid highly complicated analysis, we make the following simplifications that are usually true in practice to reveal the high-level performance picture: 1) the AP's rate selection algorithm will not probe a rate if its ideal throughput is lower than some other rate with measured throughput value, but will probe it otherwise, 2) $\mu_2^R[\hat{\rho}_a^{*,R}]$ and $\mu_2'^R[\hat{\rho}_a^{*,R}]$ are both close to 1 and can be approximated as 1, 3) if $\rho_a^{*,R}$ is not probed and is lower than any probed rate, the PRR from the AP to N at any probed rate is very small and can be approximated as 0, and 4) the links between R and N are better than those between the AP and N.

Remark 1: The approximate performance ratio with respect to AP rate selection is $\frac{3}{2}$.

Explanation: We note that if the AP has probed $\rho_a^{*,R}$, R should have set $\hat{\rho}_a^{*,R}$ to be $\rho_a^{*,R}$. Therefore we consider when the AP has not probed $\rho_a^{*,R}$. We note that $\rho_a^{*,R}$ is unlikely to be higher than $\hat{\rho}_a^{*,R}$, because the AP's rate selection algorithm should probe rates higher than $\hat{\rho}_a^{*,R}$ given that it sees good performance at $\hat{\rho}_a^{*,R}$, and R should have samples at higher rates and find that higher rates lead to better performance. Therefore, $\rho_a^{*,R}$ must be one of the lower data rates.

We denote the packet delivery time from the AP to N at rate $\rho_a^{*,R}$ with the help of R as Ψ. According to our assumption, the packet delivery time from R to N should be no more than Ψ. We also claim that the packet delivery time from the AP to R should be no more than $\frac{\Psi}{2}$. This is because according to our policy, only retransmitted packets are relayed, such that the AP will probe rates as low as $\frac{\hat{\rho}_a^{*,R}}{2}$; as a result, the optimal rate is no higher than $\frac{\hat{\rho}_a^{*,R}}{2}$. The ratio is therefore $\frac{3}{2}$. ■

I. Relayer Selection

In case multiple relayers are deployed in the same network, the best relayer should act as the relayer of N. We explain in the following how the relayers coordinate between each other to achieve this.

1) High Level Description: The high level idea of relayer selection is very simple. We denote the relayer of N as R. In our current implementation, every 1 second, R announces a query message containing its rank. If another relayer Q finds it has a better rank than R in the last 3 announcements, it will respond the query with its own rank. After a timeout, if R finds that it has received one or multiple responses to its query, it will find the best rank in the responses which, without loss of generality, supposedly is from Q. R will announce a message to let Q be the new relayer.

2) Reliable Communication with ACK-Reflection: The major challenge in relayer selection is to ensure that the control messages such as the query and the query response are received reliably, even when the relayers may not be able to communicate with each other directly. This is achieved by a trick we call *ACK reflection*, which is based on the fact that if R and Q are both potential relayers of N, they should at least be able to communicate with N reliably. As the same procedure is followed by all relayers, we explain ACK reflection in the following when R is sending the message.

Basically, R will send two very small dummy data packets back to back to N at the lowest rate and the information is announced by modifying the source MAC addresses in the packets. According to 802.11, N will send an ACK after it has received any data packet correctly, echoing the source MAC address in the data packet. Therefore, when Q receives the ACKs, it can decode the information based on the MAC addresses in the ACKs. This approach is very reliable because the dummy packets are sent at the lowest rate and the triggered ACKs are also likely at the lowest rate. The overhead is small because the dummy packets are small.

With two packets, the MAC address field carries a total of 12 bytes, among which 6 bytes are used to specify the

MAC address of N, 4 bytes the MAC address of R, and the remaining 2 bytes the message type and the rank of the relayer. The message types include query and query response, as well as relayer status announcement, which is used by a relayer to announce who is the relayer of N. We note that 4 bytes are sufficient to specify the MAC address of the relayer because the relayers are likely made by the same manufacturer sharing at least two bytes in their MAC addresses. The duration field in the dummy packets is set to a special value such that the duration field of the ACK is also a special value to simplify the detection of the ACK reflection messages.

3) Learning the Existence of N: We argue that it is unlikely that R is completely unaware of N while R is actually the best relayer. We note that R cannot receive any packet with the MAC address of N only if all such packets are at data rates too high to be received by R. If N does not have a relayer, i.e., communicates with the AP directly, R is not a good relayer. If N has a relayer Q, Q should enter rate exploration periodically and some downlink packets to N might be sent at lower data rates. If R cannot receive any such packets, the link between R and the AP is much worse than that between Q and the AP. In this case, if R is the better relayer, the link between R and N must be better than that between Q and N. However, N will transmit some uplink packets at least occasionally, and R should have learned about N from such packets.

4) Analysis: We give an approximate analysis on relayer selection. One potential problem in relayer selection, which arises because of the existence of multiple data rates, is that the AP may be "hijacked" by a relayer R such that it only transmits at rates preferred by R which do not include $\rho_a^{*,Q}$ which is the optimal rate preferred by a better relayer Q. As a result, Q may never realize that it is actually a better relayer. We define the performance ratio with respect to relayer selection as the largest possible $\frac{K_R}{K_Q}$, i.e., the ratio of the actual rank of R over that of Q, where Q is the optimal relayer. We make the following simplifying assumptions which are usually true in practice to avoid highly complicated analysis: 1) by probing, Q has learned Λ and Υ which are the delivery time of a unit size packet between the AP and itself at $\rho_a^{*,Q}$ and between Q and N at an optimal rate, respectively, 2) $\mu_2^Q[\rho_a^{*,Q}] = 1$ if $\mu_1[\rho_a^{*,Q}] > \frac{1}{2}$, 3) $K_Q = \Lambda + \Upsilon - \mu_1[\rho_a^{*,Q}]\Upsilon$, 4) ACK can always be received if the preceding packet has been received, and 5) the links between potential relayers of N and N are better than those between the AP and N.

Remark 2: The approximate performance ratio with respect to relayer selection is 2.

Explanation: We note that if $\mu_1[\rho_a^{*,Q}] \leq \frac{1}{2}$, $K_Q \geq \Lambda + \frac{\Upsilon}{2}$. On the other hand, Q must have evaluated $\Lambda + \Upsilon$ as an upper bound of its rank and found it larger than the announced rank of R, which is no less than K_R. Therefore,

$$\frac{K_R}{K_Q} \leq \frac{\Lambda + \Upsilon}{\Lambda + \frac{\Upsilon}{2}} \leq \frac{\Lambda + \Upsilon}{\frac{\Lambda}{2} + \frac{\Upsilon}{2}} = 2.$$

If $\mu_1[\rho_a^{*,Q}] > \frac{1}{2}$,

$$\frac{K_R}{K_Q} \leq \frac{\Lambda + \Upsilon}{\Lambda + \Upsilon - \mu_1[\rho_a^{*,Q}]\Upsilon} = 1 + \frac{\mu_1[\rho_a^{*,Q}]\Upsilon}{\Lambda + \Upsilon - \mu_1[\rho_a^{*,Q}]\Upsilon}.$$

According to our assumption, $\frac{1}{\rho_a^{*,Q}\mu_1[\rho_a^{*,Q}]} \geq \Upsilon$. Also, as we assume $\mu_2^Q[\rho_a^{*,Q}] = 1$ when $\mu_1[\rho_a^{*,Q}] > \frac{1}{2}$, $\Lambda = \frac{1}{\rho_a^{*,Q}} \geq \mu_1[\rho_a^{*,Q}]\Upsilon$. Therefore,

$$\frac{K_R}{K_Q} \leq 1 + \frac{\mu_1[\rho_a^{*,Q}]\Upsilon}{\mu_1[\rho_a^{*,Q}]\Upsilon + \Upsilon - \mu_1[\rho_a^{*,Q}]\Upsilon} = 1 + \mu_1[\rho_a^{*,Q}] \leq 2$$

because $\mu_1[\rho_a^{*,Q}] \leq 1$. ■

IV. Evaluations

We implement the proposed L2Relay protocol on the OpenFWWF platform [15] in firmware. We conduct our evaluation in emulated wireless channel conditions as well as in real-world wireless channel conditions comparing against a commercial range extender.

A. Implementation

In the OpenFWWF platform, programs are written in assembly language and are loaded into the microprocessor in the BCM4306 card as the firmware. The program has control over carrier sense, microsecond level timing, and the access to many physical layer parameters. We added about 2,000 lines of code to the original firmware. Our current implementation is a complete implementation of the L2Relay protocol with all features. The only limitation is due to the limited memory size of the microprocessor which cannot store many PRR tables; our current implementation supports only the 802.11g rates and each relayer can support up to 4 nodes.

B. Evaluation in Emulated Environments

Our first set of evaluations is based on emulated wireless channels. We install a packet dropping module which drops the packets and ACKs according to the emulated channel conditions. The advantage of emulation-based evaluation is that the channel is repeatable and theoretically predictable. Real wireless channels are subject to random changes caused by interference and fading, such that it is often difficult to validate the internal designs of the protocol.

1) The Emulated Channel: In the emulation, we set the AP, the node, and the relayers to be physically very close to each other such that natural packet losses are rare and most losses are introduced by the dropping module. The PRRs of the links are determined by the emulated Signal to Noise Ratio (SNR) and the SNR to PRR table of the BCM4306 card available at [17]. The SNR is determined by the distance in the emulated environment between the sender and the receiver; to be specific, if the distance is d measured in meters, the received signal strength is assumed to be $-31 - 30\log_{10} d$ measured in dBm according to the path loss model at [16].

2) Compared Schemes: We compare L2Relay with two other schemes: 1) "Baseline," which has only the AP and the node, and 2) "Emulated Range Extender (EmuRExt)," in which an emulated range extender is deployed in addition to the AP and the node. EmuRExt is also implemented in the OpenFWWF platform and emulates the behavior of typical range extenders which repeat every packet. That is, whenever EmuRExt receives a packet from the AP, it sends an ACK to the AP and relays the packet SIFS after the ACK. We hack the firmware of the node such that it never sends ACK to the AP when running EmuRExt experiments. EmuRExt uses the same algorithm as L2Relay for rate selection to the node.

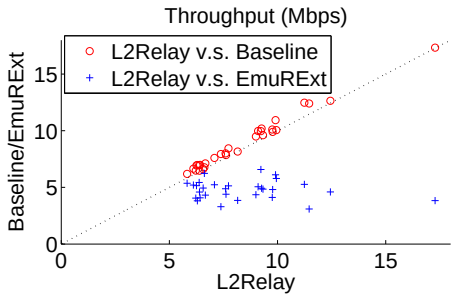

Fig. 3. Scatter plot of the throughput in the "one-hop" case.

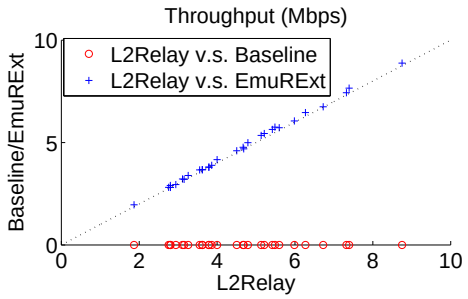

Fig. 4. Scatter plot of the throughput in the "two-hop" case.

3) The Experiment Setup: All our experiments involve one AP, one node, and one or three relayers on 802.11g channel 3. The AP, the node, and the relayers are Dell 610 laptop computers with the BCM4306 card. The AP and the node run the unmodified b43 driver with auto-rate enabled and slightly modified OpenFWWF firmware with the additional packet dropping module as described earlier. The relayer functions in the firmware level and need driver no support; the L2Relay firmware or the EmuRExt firmware is loaded to turn it into a L2Relay relayer or an EmuRExt relayer, along with the additional packet dropping module. The AP is always at the center of the emulated environment. The locations of the relayers and the node are randomly selected within 20 to 150 meters to the AP under certain addition constraint; to be more specific, random locations are selected first but the selections may be rejected and the process repeated until the constraint is satisfied. A separate machine is used as a sniffer to collect log files with TCPDump. The iperf [14] tool is used to generate UDP traffic as well as reporting the link throughput values. All experiments run for 40 seconds and the data is collected from 10 second to 40 second.

4) Performance Comparison: We first compare the performance of the schemes when there is only one EmuRExt or L2Relay relayer. In the experiments, the AP is at the center of the emulated environment, while the relayer and the node are at random locations under certain additional constraints depending on the experiment.

The first type of experiments is referred to as "one-hop," i.e., when there is no need to relay. The relayer and node locations are randomly chosen under the constraint that the node will not see improved performance with the addition of the L2Relay relayer. Fig. 3 shows the scatter plot of the throughput collected in 30 experiments. In the scatter plot, the x and y coordinates of a marker are the throughput values of L2Relay and the compared scheme in one experiment, respectively. We can see that L2Relay achieves almost identical performance as Baseline, which is the desired behavior because there is no need to relay packets in this case. EmuRExt suffers much worse performance because it mechanically rebroadcasts every packet; in some cases, it uses a much lower rate than the rate of the AP and significantly reduces the throughput.

The second type of experiments is referred to as "two-hop," i.e., when relay is absolutely needed, by choosing random relayer and node locations under the constraint that the node cannot communicate with the AP even at the lowest data rate. Fig. 4 shows the scatter plot of the throughput collected in 30 experiments. We can see that L2Relay achieves almost identical performance as EmuRExt, which is the desired behavior because EmuRExt is optimal in the two-hop setting. Clearly, Baseline will report a throughput of 0.

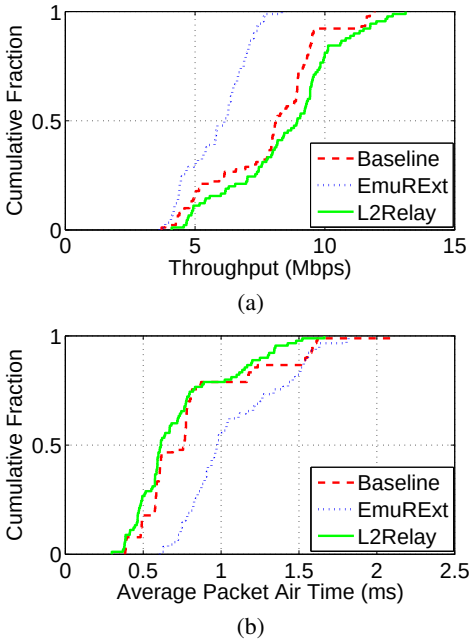

Fig. 5. The "middle-ground" case. (a) CDF of the throughput. (b). CDF of the packet air time.

The third type of experiments is referred to as "middle-ground," i.e., the relayer and node locations are chosen under the constraint that the AP can communicate with the node directly while adding an L2Relay relayer will theoretically improve the throughput. Fig. 5(a) shows the Cumulative Density Function (CDF) of the throughput collected in 90 experiments. We can see that L2Relay achieves noticeable throughput gains than the other two schemes. For example, the median throughput gain of L2Relay over Baseline and EmuRExt are 10% and 45%, respectively. The reasons of the throughput gain in middle-ground experiments are not as obvious as the other two cases. It may be because the AP may not need to retransmit a packet as the retransmission is handled more efficiently by the relayer which shares a better link with the node. It may also be because the AP may use higher data rates. We measure the average air time consumption to deliver a packet based on the log file, including the transmissions from both the AP and the relayer, and show the CDF in Fig. 5(b). We can see that it cross verifies with Fig. 5(a) which confirms that the throughput gain is achieved by more efficient packet forwarding.

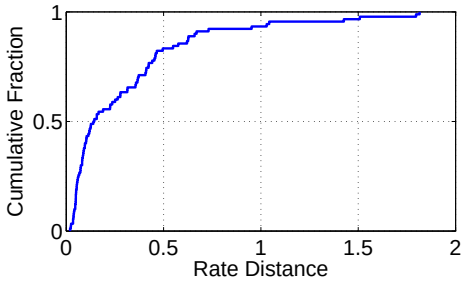

Fig. 6. CDF of the rate distance.

5) Rate Selection: A key issue in L2Relay is the rate selection of the AP because the relayer should assist the AP to settle at a good rate even when the AP is completely unaware of the relayer. We denote rate i as ρ_i where $1 \leq i \leq 8$ because there are 8 rates in 802.11g. We define the *rate distance* as $\sum_{i=1}^{8} w_i|i-j|$, where w_i is the fraction of packets sent at rate ρ_i and ρ_j is the optimal rate calculated according to the PRRs in the emulated environment. Clearly, the rate distance is a metric of the rate selection; for example, a rate distance of 1 may mean that all packets are sent at a rate next to the optimal rate, and a rate distance of 0.5 may mean that half of the packets are sent at the optimal rate while the other half at a rate next to the optimal rate. Fig 6 shows the CDF of the rate distance in the middle-ground case, where we can see that rate distance is typically small with a median of around 0.15, which means that the AP uses the optimal rate or rates close to the optimal rate in the majority of the cases.

6) Relayer Selection: We conduct separate experiments to test relayer selection which is also a key aspect of L2Relay. As before, the AP is at the center of the emulated environment. Three L2Relay relayers are randomly located on two circles corresponding to maximum communication speeds at 36 Mbps and 24 Mbps, respectively. The location of the node is randomly selected under the constraint that at least one of the relayers may act as the relayer of the node. A total of 50 sets of experiments are conducted, where each set contains two types of experiments, referred to as "L2Relay" and "Optimal." In L2Relay, all 3 relayers are turned on; in Optimal, only the optimal relayer calculated according to the PRRs in the emulated environment is turned on. To evaluate the speed the optimal relayer captures the relayer role, in the L2Relay case, the optimal relayer is turned on 5 seconds after other relayers. Fig. 7(a) shows the scatter plot of the throughput, where we can see that the throughputs of L2Relay and Optimal are almost identical, which confirms that L2Relay is capable of converging to a good relayer in most cases. In all our experiments, we find that the optimal relayer eventually becomes the relayer of the node and stays as the relayer of the node afterwards till the end of the experiment. Fig. 7(b) shows the CDF of the activation time of the optimal relayer, defined as the time since the optimal relayer is turned on to the time it becomes the relayer of the node. We can see that the activation time is reasonably fast with a median of around 4 seconds.

C. Evaluation in Real-world Environments

We also conduct experiments in real-world environments comparing head-to-head with the Netgear extender [1]. A total of 15 network topologies are experimented, where each

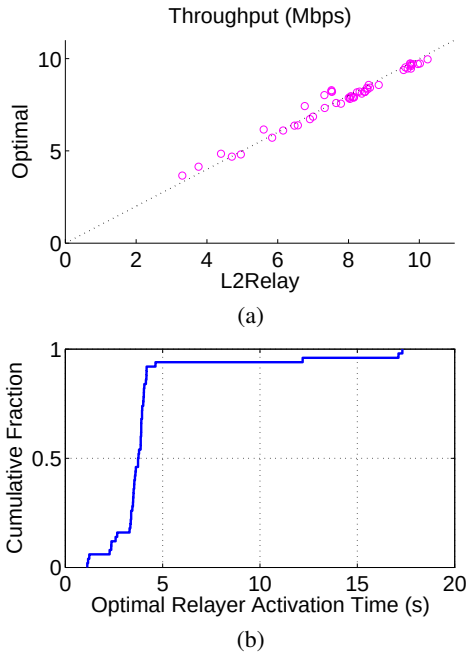

Fig. 7. Relayer selection. (a). Scatter plot of throughput. (b). CDF of the activation time of the optimal relayer.

topology has one AP, 4 nodes, 3 L2Relay relayers, and one Netgear extender. In each topology, there are both nodes close to the AP and far from the AP; the L2Relay relayers and the Netgear extender are typically in the middle range. Only one Netgear extender is used because only one can be deployed in a network.

For each topology, we conduct the "one-to-one" experiments in which the AP runs only one iperf session and transmits to only one node, as well as the "one-to-many" experiments in which the AP runs 4 iperf sessions and transmits to all 4 nodes. We use "Baseline" to refer to experiments where only the AP and the node(s) are turned on. We use "Netgear" to refer to the experiments in which the Netgear extender is turned on in addition to the AP and the node(s). To be fair, we actually try 3 times, placing the Netgear extender at the same location as each of the L2Relay relayers. For one-to-one experiments, the node throughput is simply the highest among the three attempts. For one-to-many experiments, the node throughput is the one from the attempt with the highest aggregate throughput. We use "L2Relay" to refer to the experiments in which all 3 L2Relay relayers are turned on in addition to the AP and the node(s).

Fig. 8(a) shows the scatter plot of the throughput in the one-to-one experiments. We note that the comparison between L2Relay and Netgear may actually be biased against L2Relay, because the Netgear extender has better hardware such as stronger antenna. Still, L2Relay achieves higher throughput than both Baseline and Netgear in most cases, which confirms that L2Relay is effective in real-world environments. The throughput of Baseline is higher than Netgear in many cases when the nodes are close to the AP and do not need any packet repeating. The Netgear extender almost delivers the same throughput regardless of the location of the nodes, which is due to its static behavior of repeating every packet, as well as its strong hardware capable of supporting similarly high data rates to all nodes in the experiments.

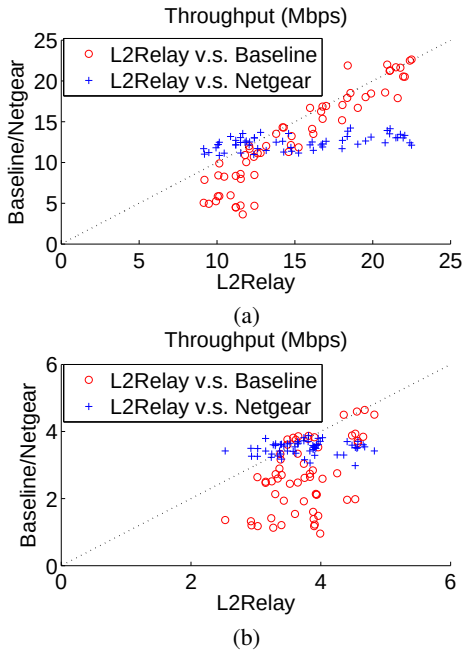

Fig. 8. Scatter plot of throughput in the real-world experiments. (a) One-to-one. (b). One-to-many.

Fig. 8(b) shows the scatter plot of the throughput in the one-to-many experiments. We can see that L2Relay still outperforms both Baseline and Netgear in most cases, which confirms that L2Relay is capable of supporting multiple nodes effectively. The gain over Netgear is not as impressive as in the one-to-one case because of the fate sharing in 802.11 networks; that is, a slow node may consume much of the air time with its traffic thus dragging down the performance of all other nodes. As explained earlier, the Netgear extender can deliver almost the same performance to all nodes, thus there are practically no slow nodes with Netgear. On the other hand, some nodes will be slower with L2Relay than with Netgear due to the better hardware of Netgear, which will negatively impact the performance of all nodes. Also due to the fate sharing, Baseline sees a much weaker performance, because a slow node becomes even slower when deprived of the help from any relaying devices. Finally, we note that the main purpose of the one-to-many experiments is to confirm the effectiveness of L2Relay in supporting multiple nodes; the fate sharing problem should be solved by better link bandwidth sharing algorithms which are out of the scope of this paper.

V. Conclusion

In this paper, we propose a novel layer 2 packet relay protocol for Wi-Fi called L2Relay. An L2Relay relayer overhears a lost packet and retransmits the packet on behalf of the original sender if it is in a good position to relay the packet. L2Relay is designed to be ubiquitously compatible, i.e., one or multiple relayers can be easily installed in any network without any modification to the AP or the nodes. We solve problems such as link quality estimation with passive and active measurements, relayer rank calculation, rate adaptation for the sender node, and distributed relayer selection. Our solutions exploit many layer 2 functionalities such as carrier sense and microsecond level timing. We implement L2Relay in the OpenFWWF platform and compare it against the baseline without a relayer as well as a commercial Wi-Fi range extender. Our results show that L2Relay outperforms both compared schemes.

Acknowledgements

This research work was supported in part by the US National Science Foundation under Grant 1149344.

References

[1] Netgear Universal Wi-Fi Range Extender WN3000RP-100NAS, *http://www.netgear.com/home/products/wireless-range-extenders/wn3000rp.aspx*.

[2] P. Bahl, R. Chandra, P. Lee, V. Misra, J. Padhye, D. Rubenstein, and Y. Yu, "Opportunistic use of client repeaters to improve performance of WLANs," in *ACM CoNEXT*, 2008.

[3] M.-H. Lu, P. Steenkiste, and T. Chen, "Design, implementation and evaluation of an efficient opportunistic retransmission protocol," in *ACM Mobicom*, 2009.

[4] D.-W. Kim, W.-S. Lim and Y.-J. Suh, "A robust and cooperative MAC protocol for IEEE 802.11a wireless networks," in *Wireless Pers Commun*, DOI 10.1007/s11277-011-0405-5, 2011.

[5] B. Hagelstein, M. Abolhasan, D. Franklin and F. Safaei, "An efficient opportunistic cooperative diversity protocol for IEEE 802.11 networks," in *International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 417-421, Caen, France, 2010.

[6] H. Zhu and G. Cao, "rDCF: A relay-enabled medium access control protocol for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol.5, no. 9, pp. 1201-1214, 2006.

[7] S. Zou, B. Li, H. Wu, Q. Zhang, W. Zhu, and S. Cheng, "A relay-aided media access (RAMA) protocol in multirate wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 9, pp. 1657-1667, 2006.

[8] P. Liu, Z. Tao, S. Narayanan, T. Korakis, and S.S. Panwar, "CoopMAC: A cooperative MAC for wireless LANs," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 2, pp. 340-354, 2007.

[9] J. Bicket, "Bit-rate selection in wireless networks," *PhD thesis, Massachusetts Institute of Technology*, 2005.

[10] D. Halperin, W. Hu, A. Sheth, and D. Wetherall, "Predictable 802.11 packet delivery from wireless channel measurements," in *ACM SIGCOMM*, 2010.

[11] G. Judd, X. Wang, and P. Steenkiste, "Efficient channel-aware rate adaptation in dynamic environments," in *ACM MobiSys*, 2008.

[12] S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," in *ACM MobiCom*, 2006.

[13] B. Han, A, Schulman, F. Gringoli, N. Spring, B. Bhattacharjee, L. Nava, L. Ji, S. Lee, and R. Miller, "Maranello: Practical partial packet recovery for 802.11," in *NSDI, 2010*.

[14] iperf, *http://iperf.sourceforge.net/*.

[15] F. Gringoli and L. Nava, "Open FirmWare for WiFi networks: a UniBS NTW group project," *http://www.ing.unibs.it/~openfwwf/*.

[16] A. Bose and C. H. Foh, "A practical path loss model for indoor WiFi positioning enhancement," in *ICICS*, Singapore, December 2007.

[17] "Broadcom WLAN chipset for 802.11abg," http://www.docstoc.com/docs/53462044/Broadcom-WLAN-Chipset-for-80211abg.

[18] "Gnu radio - gnu fsf project," *http://www.gnu.org/software/gnuradio.*