Greedy Receivers in IEEE 802.11 Hotspots: Impacts and Detection

Mi Kyung Han and Lili Qiu
Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712
{hanmi2,lili}@cs.utexas.edu

***Abstract*— As wireless hotspot business becomes a tremendous financial success, users of these networks have increasing motives to misbehave in order to obtain more bandwidth at the expense of other users. Such misbehaviors threaten the performance and availability of hotspot networks, and have recently attracted increasing research attention. However the existing work so far focuses on sender-side misbehavior. Motivated by the observation that many hotspot users receive more traffic than they send, we study greedy receivers in this paper. We identify a range of greedy receiver misbehaviors, and quantify their damage using both simulation and testbed experiments. Our results show that even though greedy receivers do not directly control data transmission, they can still result in very serious damage, including completely shutting off the competing traffic. To address the issues, we further develop techniques to detect and mitigate greedy receiver misbehavior, and demonstrate their effectiveness.**

Keywords: C.2.1.k [Wireless Communication], C.2.5 [Local-Area Networks].

I. Introduction

The proliferation of lightweight hand-held devices with built-in high-speed WiFi network cards has spurred widespread deployment of wireless "hot-spot" networks at many public places, such as hotels, airports, restaurants, and malls. As reported in [8], [9], worldwide wireless data hotspot revenue will rise from $969 million in 2005 to $3.46 billion in 2009, and the number of hotspot locations will nearly double in size from 100,000 in 2005 to almost 200,000 by the end of 2009. As hotspot business becomes a tremendous financial success, users of these networks have increasing incentives to misbehave in order to gain more bandwidth even at the expense of others.

The serious damage caused by MAC-layer misbehavior has recently received substantial research attention. Some of the pioneering work in this area includes [2], [12], [14], [15]. These works identify several types of MAC-layer misbehaviors, and propose countermeasures to detect and prevent such misuse.

The existing work so far focuses on sender-side misbehavior. In wireless LAN (WLAN) networks, the amount of traffic coming from access points (APs) to clients is typically higher than that from clients to APs [10], [19]. APs are under the control of service providers and send more data, whereas (possibly misbehaving) users often act as receivers. Therefore misbehaving receivers can be serious threats to the performance and availability of WLANs. However, there is little work on receiver-side MAC misbehaviors. This motivates our work.

In this paper, we first identify a range of greedy receiver misbehaviors. Such receiver misbehaviors are possible because IEEE 802.11 is a feedback-based protocol; while receivers do not directly control data transmissions, they can cause damage by manipulating the feedback. The broadcast nature of wireless medium makes it easy to manipulate not only its own feedback but also other flows' feedback. We quantify the performance impact of misbehaving receivers using both simulation and testbed experiments. Our results show that misbehaving receivers can cause serious damage to the network. In some cases, a greedy receiver can completely shut off the other competing flows. To mitigate the threats and enhance network availability, we further develop techniques to detect and mitigate greedy receiver misbehavior, and demonstrate their effectiveness.

The rest of the paper is organized as follows. We overview the background of IEEE 802.11 and TCP in Section II, and survey related work in Section III. We present a range of greedy receiver misbehaviors in Section IV. We quantify their damage using simulation and testbed experiments in Section V and Section VI, respectively. We describe techniques to detect and mitigate greedy receiver misbehavior in Section VII, and evaluate its effectiveness in Section VIII. We conclude in Section IX.

II. Background

IEEE 802.11: The IEEE 802.11 standard [5] specifies two types of coordination functions for stations to access the wireless medium: distributed coordination function (DCF) and point coordination function (PCF). In this paper, we focus on DCF, which is much more widely used than PCF. In IEEE 802.11 DCF [5], before transmission, a station first checks to see if the medium is available using virtual carrier-sensing and physical carrier-sensing. The medium is considered busy if either carrier-sensing indicates so.

Virtual carrier-sensing is performed using the Network Allocation Vector (NAV). Most 802.11 frames have a NAV field, which indicates how long the medium is reserved in order to finish transmitting all the frames for the current operation. Virtual carrier sensing considers medium is idle if NAV is zero, otherwise it considers the medium busy. Only when NAV is zero, physical carrier-sensing is performed using carrier-sensing hardware. If physical carrier-sensing also determines the medium idle, a station may begin transmission using the following rule.

This work is supported in part by NSF Grants CNS-0546755 and CNS-0627020. An earlier version appeared in the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2007). The new version adds more extensive simulation and experimental evaluation of greedy misbehaviors and the detection schemes. We thank Brian Overstreet for help with initial ns-2 simulation.

If the medium has been idle for longer than a distributed inter-frame spacing time (DIFS), transmission can begin immediately. Otherwise, a station having data to send first waits for DIFS and then waits for a random backoff interval, which is uniformly chosen between $[0, CW_{min}]$, where CW_{min} is the minimum contention window. If at anytime during the above period the medium is sensed busy, the station freezes its counter and the countdown resumes when the medium becomes idle. When the counter decrements to zero, the node transmits the packet. If the receiver successfully receives the packet, it waits for a short inter-frame spacing time (SIFS) and then transmits an ACK frame. If the sender does not receive an ACK (e.g., due to a collision or poor channel condition), it doubles its contention window to reduce its access rate. When the contention window reaches its maximum value, denoted as CW_{max}, it stays at that value until a transmission succeeds, in which case the contention window is reset to CW_{min}.

TCP: TCP is a widely used transport protocol. One of the misbehaviors we identified exploits the interactions between TCP and MAC layer, so we briefly review TCP here. TCP provides reliable, in-order delivery service. It uses ACK, timeout, and retransmission to achieve reliability. It further provides congestion control by inferring congestion based on packet losses. Upon a packet loss as indicated by receipt of 3 duplicate ACKs or timeout, the TCP sender assumes the network is under congestion and responds by reducing its congestion window (*i.e.*, the maximum amount of unacknowledged data allowed by the TCP sender). Since not all wireless losses are due to congestion, TCP congestion control is sometimes unnecessary and can cause performance degradation. One way to address this issue is to hide wireless losses from TCP via local retransmission, thereby avoiding unnecessarily reducing sending rate. This is the approach taken by IEEE 802.11.

III. Related Work

The serious damage caused by misbehaving MAC has received increasing attention in wireless research community. For example, Bellardo and Savage [2] studied denial of service attacks in 802.11. Kyasanur and Vaidya [12] identified that selfish senders can get significantly more bandwidth than regular senders by modifying the backoff value in IEEE 802.11. Raya et al. [14], [15] developed DOMINO, a software installed on access points to detect and identify greedy stations. Guang [6] developed a Predictable Random Backoff (PRB) algorithm to force each node to generate a predictable backoff interval and detect hosts that do not follow the protocol. Cagalj et al. [3] used a game-theoretic approach to study selfish nodes in CSMA/CA networks. Unlike the existing work, which focuses on sender-side misbehavior, we identify a range of receiver-side misbehaviors and evaluate their impact on network performance.

In addition to MAC misbehaviors, researchers also studied misbehavior at other protocol layers, such as jamming attacks [20], routing attacks [7], and selfish TCP behavior and attacks [1], [11], [17]. In particular, our work is inspired by [17], which studies TCP receiver misbehaviors and shows that in a feedback-based protocol receivers can significantly affect network performance even though they do not directly send data. Unlike [17], we study receiver misbehavior at MAC-layer.

IV. Greedy Receiver

In this section, we present three types of greedy receiver misbehaviors. Among them, two misbehaviors – spoofing ACKs and sending fake ACKs are new. The other misbehavior – NAV inflation has been identified earlier. Our work complements the previous studies on NAV inflation [2], [14], [15] in the following aspects. First, we focus on greedy receivers (*i.e.*, the frames with inflation can only be transmitted by receivers). In comparison, [14], [15] focuses on greedy senders and [2] focuses on denial-of-service (DOS) attacks, where misbehaving nodes simply cause damage without necessarily gaining more throughput. We will show that only a small NAV increase is required for GR to starve other flows due to additional data traffic, whereas a large NAV inflation is required to launch the type of DOS considered in [2]. Second, we present a simple analysis to model the effect of NAV inflation in Section V. Third, we will use extensive evaluation to study the effect of NAV inflation in various scenarios (*e.g.*, studying the impact of the amount of inflation, the type of frames with inflation, the frequency of inflation, the number of competing receivers, and the type of the transport protocols, etc.).

For each misbehavior, we first introduce the misbehavior and then describe its applicable scenarios, greedy actions, and effects. Throughout the paper, we let GR denote a greedy receiver, NR denote a normal receiver, GS denote GR's sender, and NS denote NR's sender. We assume that APs are the senders and behave normally, since they are under the control of service providers.

A. Misbehavior 1: Increasing NAV

IEEE 802.11 uses NAV to perform virtual carrier sensing. Greedy receivers can increase their goodput (i.e., rate of correctly received non-duplicate packets) by increasing NAV.

Applicable Scenarios The misbehavior is effective whenever there is traffic competing with a greedy receiver. Inflated CTS NAV causes damage only when RTS/CTS is enabled, whereas inflated ACK NAV causes damage regardless whether RTS/CTS is used. When TCP is used, the greedy receiver also sends TCP ACK packets, which are data frames to the MAC layer. As a result, the greedy receiver can also inflate NAV on the RTS and data frames, which are used to send the TCP ACK packet.

Greedy Actions A greedy receiver may inflate NAV in its CTS and/or ACK frames under UDP, and inflate NAV in CTS, ACK, RTS, and/or data frames under TCP. It can increase the NAV up to $32767\mu s$, which is the maximum allowable value in IEEE 802.11.

Effects Sending frames with inflated NAV allows a greedy receiver to silence all nearby nodes longer than necessary. According to IEEE 802.11 [5], upon receiving a valid frame,

each station should update its NAV, only when the new NAV value is greater than the current NAV value and only when the frame is not addressed to the receiving station. Thus the increased NAV value will not affect GS, which sends data to GR, but silence the other nearby senders and receivers.

If the amount of NAV increase is large enough, GS can exclusively grab the channel even in presence of other nearby competing senders since it always senses the medium idle before its transmission.

B. Misbehavior 2: Spoofing ACKs

Upon a packet loss, a TCP sender reduces its sending rate by decreasing its congestion window. MAC-layer retransmissions help to reduce packet losses observed at the TCP layer. Based on the observation, a greedy receiver can send MAC-layer ACKs on behalf of other TCP flows. In this way, packet losses are not recovered at MAC-layer as they should, but are propagated to the TCP layer, which can cause TCP senders to slow down.

Applicable Scenarios The misbehavior is effective under the following two conditions. First, the traffic competing with greedy receiver is TCP and its link is lossy. Second, a greedy receiver uses promiscuous mode so that it can spoof MAC-layer ACKs in response to data frames not destined to itself.

Greedy Actions A greedy receiver (GR) sniffs a data frame destined to a normal receiver (NR) coming from a sender (NS), and sends a MAC-layer ACK on behalf of NR. Because the link from NS to NR is lossy, NR may not successfully receive the data. However GR spoofs a MAC-layer ACK on behalf of NR so that NS moves on to the next transmission, instead of performing MAC-layer retransmissions as it should.

Effects In order to understand the effects of a spoof ACK, we need to consider two cases. The first case is that the original receiver (NR) does not receive the data frame so the spoofed ACK from GR effectively disables MAC-layer retransmission at NS. This propagates packet losses to NS's TCP, which will decrease its congestion window and may even cause TCP timeouts, thereby increasing the traffic rate towards the greedy receiver. When the normal traffic spans both wireless and wireline network, the damage of this misbehavior is further increased; The additional wireline delay makes end-to-end TCP loss recovery even more expensive than local MAC-layer retransmissions on the wireless link. We also observe this effect in our evaluation, as described in Section V.

Second, when NR receives the data frame, spoofed ACK will collide with the ACK from the original receiver NR. Such collisions cause unnecessary retransmissions from NS and slow down NR's flow. This is essentially a jamming attack, which has been studied before (*e.g.,* [20]). Therefore our evaluation focuses on the first case – disabling MAC-layer retransmissions. The performance degradation caused by greedy receiver would be even larger under the combination of jamming and disabled MAC retransmissions.

One way to focus on the first case is to let the greedy receiver only spoofs ACKs when the normal receiver does not correctly receive the data frame. But in practice the greedy receiver does not have such information. Therefore in order to focus on the first case, our evaluation considers capture effects so that there is no collision even if both receivers send ACKs. Specifically, when the two packets are received simultaneously, if the ratio of their received signal strength is above capture threshold, only the packet with stronger signal is received and the other is lost. In our context, we consider either $RSS_{NR}/RSS_{GR} \geq Thresh_{cap}$ or $RSS_{GR}/RSS_{NR} \geq Thresh_{cap}$, where $Thresh_{cap}$ is capture threshold, and RSS_{NR} and RSS_{GR} are received signal strength from NR and GR, respectively. In the former case, ACK from NR is demodulated and received, and ACK from GR is lost, and in the latter case, the ACK from GR is received and the ACK from NR is lost.

Remarks: In addition to degrading the performance of competing TCP traffic that experiences packet losses, spoofing ACKs may also negatively impact certain competing UDP traffic. In particular, the misbehavior is effective when the competing UDP flow does not send more traffic to the MAC layer as the result of packet losses. In this way, disabling MAC-layer retransmission at the normal receiver translates to reduced service time for the normal receiver and increased service time for the greedy receiver. An example scenario is that an AP sends traffic to both greedy and normal receivers and the sending rate to the normal receiver at the transport layer does not change in response to disabling MAC retransmissions.

If the competing flow tries to recover the packet losses at the high-layer or sends other data traffic in place of retransmissions (*e.g.*, an AP sends traffic to a normal receiver as fast as possible and competes with another AP that sends traffic to a greedy receiver), then disabling MAC-layer retransmission does not reduce the effective service time to the normal receiver so the greedy receiver no longer benefits.

Since ACK spoofing degrades the performance of competing TCP traffic in all cases and also causes more damage, our evaluation mostly focuses on TCP traffic. But for completeness we also study its effect on UDP traffic.

C. Misbehavior 3: Sending fake ACKs

In 802.11, a sender performs an exponential backoff upon seeing a packet loss. This slows down the sender when network is congested and packets get corrupted. A greedy receiver can prevent its sender from backing off by sending ACKs even when receiving corrupted packets (destined to itself). In this way, the greedy receiver receives a higher goodput (*i.e.*, the receiving rate of uncorrupted and unduplicate packets).

Applicable Scenarios This misbehavior is effective under the following three conditions. First, the link from GS to GR is lossy. Second, the traffic to GR is carried by non-TCP connections (to avoid interacting with TCP congestion control). Third, the greedy flow either tries to recover the packet loss at a high layer or sends other data traffic in place of retransmissions so that removing MAC layer retransmissions does not result in reduced service time to the greedy receiver.

Greedy Actions When receiving a corrupted frame, GR sends a MAC-layer ACK back to the source even though the data is actually corrupted. Moreover, to avoid reducing

	# received	# corrupted	# corrupted w/ correct dest	# corrupted w/ correct src-dest
802.11b	65536	1367	1351	1282
802.11a	23068	7376	6197	5663

TABLE I

TESTBED MEASUREMENT SHOWS THAT MOST CORRUPTED PACKETS PRESERVE SOURCE AND DESTINATION MAC ADDRESSES.

its service time, the greedy flow either tries to recover the lost packets at the high layer or sends other data packets to the MAC layer. For example, a greedy flow sends as fast as possible when it competes with another flow from a different AP. In this case, even though the greedy flow does not spend time in retransmissions, it sends out more fresh data frames.

The effectiveness of this attack depends on how often a corrupted packet preserves correct source and destination addresses. Since MAC addresses are much smaller than the payload, most of corrupted packets preserve MAC addresses. To further validate this claim, we conduct measurement experiments in our testbed by placing sender and receiver far enough to generate significant packet corruption. Table I shows a breakdown of the number of corrupted packets, corrupted packets with correct destination MAC addresses, and corrupted packets with correct source and destination MAC addresses. As it shows, 98.8% and 84% corrupted packets are delivered to the correct destination in 802.11b and 802.11a, respectively. Among them, 94.9% and 91.4% packets have correct source addresses in 802.11b and 802.11a, respectively. These numbers indicate that sending fake ACKs is a feasible attack since most of corrupted packets preserve MAC addresses.

Effects GR sending ACKs in the presence of corrupted data frames effectively prevents GS from doing exponential backoff and creates more transmission opportunity for GR, thereby increasing its goodput. An interesting aspect of this misbehavior is that it is a common belief that the link layer retransmission is considered to improve performance over end-to-end recovery; however its performance benefit can be offset by exponential backoff when competing with other flows.

Similar to misbehavior 2, misbehavior 3 also modifies how MAC-layer ACKs are transmitted under corrupted/lost packets. However, they differ in that misbehavior 2 degrades competing TCP traffic by exploiting TCP congestion control in response to packet losses and degrades competing UDP traffic by reducing its service time, whereas misbehavior 3 benefits the greedy UDP flow by avoiding MAC-layer backoff even under packet losses and hence increasing its opportunity to access the medium.

V. EVALUATION OF GREEDY RECEIVERS IN SIMULATION

In this section, we use Network Simulator 2 (NS2) [13] to quantify the damage caused by greedy receivers. We use 802.11b as the default configuration, and also evaluate 802.11a for comparison. We use 6Mbps and 11 Mbps PHY rates for 802.11a and 802.11b, respectively. Unless otherwise specified, all nodes are within communication ranges. All the senders behave normally, since APs are usually the senders that are under the control of hotspot providers and do not misbehave. We consider that NS sends to NR and GS sends to GR, where GR denotes a greedy receiver and NS, GS and NR all behave normally. Unless otherwise specified, all nodes are within the communication range, since this maximizes the effects of the attacks.

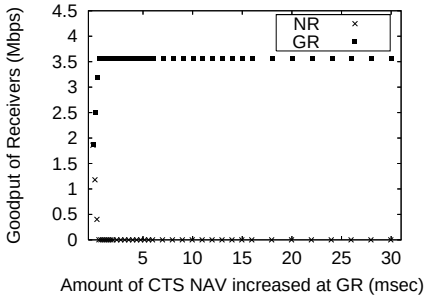


Fig. 1. Goodput of two UDP flows NS-NR and GS-GR, where GR inflates CTS NAV (802.11b)

Our evaluation uses TCP and UDP, and data packet size of 1024 bytes. When UDP is used, we generate constant bit rate (CBR) traffic high enough to saturate the medium. Moreover, the rates of all CBR flows are the same so that the difference in goodput is due to MAC-layer effect. We run each scenario 5 times and report the median of the goodput. As we will show, even though greedy receivers do not directly control data transmission, they can still effectively increase their goodput at the expense of degrading or even shutting off other competing flows.

A. Misbehavior 1: Increasing NAV

We evaluate the impact of NAV inflation by varying (i) the type of transport protocols, (ii) the amount of NAV inflation, (iii) the frequency of NAV inflation, (iv) the number of greedy receivers, and (v) the number of senders. When the greedy receiver uses UDP, it can inflate CTS and/or ACK frames. When the greedy receiver uses TCP, not only can it inflate NAV in CTS and/or ACK, but also inflate NAV in RTS and data frames when sending TCP ACKs.

Vary the amount of NAV inflation: Let n denote the original NAV value before inflation. The value of NAV used by greedy receivers is $n + \alpha \cdot 100$, where α varies from 0 to 310 for CTS NAV, and from 0 to 327 for ACK NAV. $\alpha = 310$ in CTS and $\alpha = 327$ in ACK give close to the maximum NAV, which is $32767\mu s$.

UDP traffic: First, we evaluate the impact of greedy receivers using constant-bit-rate (CBR) traffic transferred via UDP. Figure 1 shows the goodput of a normal receiver and a greedy receiver, competing with each other and both using UDP. The greedy receiver can completely grab the medium and starve the competing flow even when NAV is inflated by only 0.6 ms.

Below we analyze the effect of NAV inflation under UDP traffic. Suppose NS and GS both have an infinite amount of data to send. GR inflates NAV in either its CTS and/or ACK by v timeslots. The probability of GS transmitting in a given round is the probability that only GS sends or both GS and NS send. Let B_S denote the random backoff interval chosen by a sender S. The probability that only GS sends is $Pr[B_{GS} < B_{NS} + v - 1]$, and the probability that both GS and NS send is $Pr[B_{NS} + v - 1 \leq B_{GS} \leq B_{NS} + v + 1]$. So the probability of GS transmitting is $Pr[B_{GS} \leq$

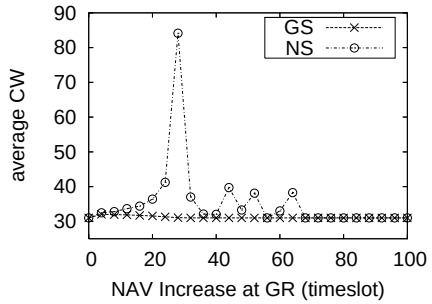


Fig. 2. Average CW of GS and NS, where two UDP flows NS-NR and GS-GR compete for the medium. (802.11b)

$B_{NS} + v + 1]$. v is added to B_{NS} because GS starts countdown v timeslots earlier than NS due to NAV inflation; and the probability that both of them send takes the above form because it takes a station 1 time slot to measure signal strength and two nodes can both send if the time of their counting down to zero differs within 1 time-slot. Similarly, the probability of NS transmitting in a round is $Prob[B_{NS} \leq B_{GS} - v + 1]$. The backoff interval is uniformly distributed over $[0..CW]$, where CW is initialized to CW_{min} and doubles every time after a failed transmission until it reaches CW_{max}.

Figure 2 shows that as NAV increases, GS's average CW stays close to the minimum CW, which lasts 31 timeslots in 802.11b, whereas NS's average CW increases. This is because NS sees an increasing fraction of collisions among the packets it sent when GS's NAV increases. (Note that while the number of collisions experiences by GS also increases, the fraction of collided packets does not increase due to an increasing number of packets GS sent.) When the NAV inflation is beyond 28 timeslots, NS sometimes cannot send even a single packet, thus its average CW remains 31 in such a case. Therefore the average CW at NS fluctuates when NAV is over 28 timeslots, depending on how many packets NS happens to send out. Based on the observation, we have the following relationship:

$$
\begin{aligned}
&Pr[GS\ sends] \\
&= Pr[B_{GS} \leq B_{NS} + v + 1] \\
&= \sum_{i=0..CW} (Pr[B_{GS} = i] \times \\
&\quad \sum_{m=CW_{min}}^{CW_{max}} Pr[CW_{NS} = m] Pr[B_{NS} \geq i - v - 1 | CW_{NS} = m]) \qquad (1)
\end{aligned}
$$

$$
\begin{aligned}
&Pr[NS\ sends] \\
&= Pr[B_{NS} \leq B_{GS} - v + 1] \\
&= \sum_{i=0..CW} (Pr[B_{GS} = i] \times \\
&\quad \sum_{m=CW_{min}}^{CW_{max}} Pr[CW_{NS} = m] Pr[B_{NS} \leq i - v + 1 | CW_{NS} = m]) \qquad (2)
\end{aligned}
$$

We evaluate the accuracy of our model by plugging the distributions of CW into Equation 1 and 2. Figure 3 compares the estimated and actual RTS sending ratios from GS and NS. As we can see, our model accurately estimates the RTS sending ratio, which is very close to the actual data sending ratio. The small difference between the two ratios is due to packet losses.

TCP traffic: Figure 4 shows TCP results using 802.11b. Figure 4(a) shows the goodput of two competing TCP flows,

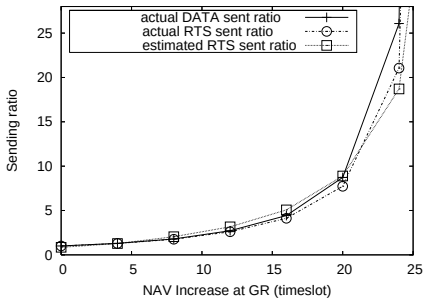
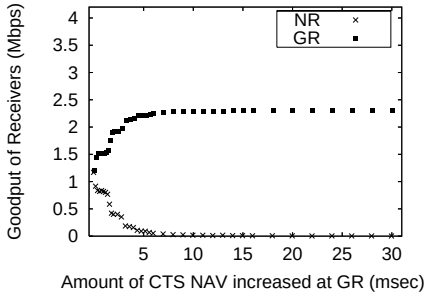
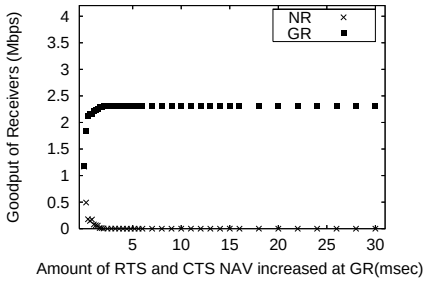


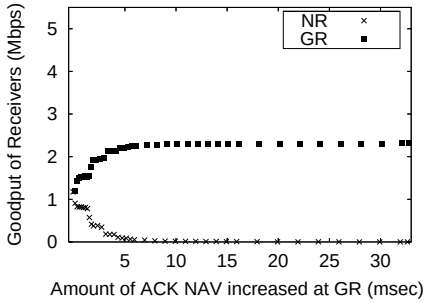
Fig. 3. Sending ratio between two competing UDP flows $GS - GR$ vs. $NS - NR$. (802.11b)



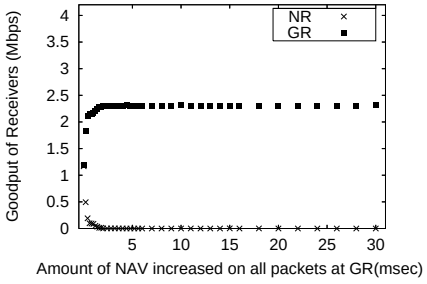
(a) *Inflated NAV in CTS*



(b) *Inflated NAV in CTS/RTS*

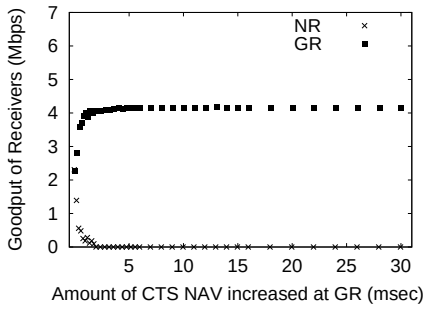


(c) *Inflated NAV in ACK*

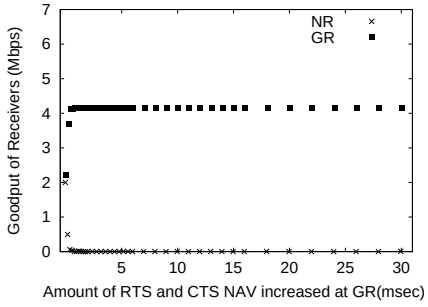


(d) *Inflated NAV in RTS/CTS/Data/ACK frames*

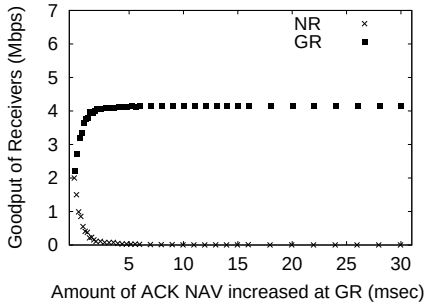
Fig. 4. Goodput of two competing TCP flows NS-NR and GS-GR for 802.11b.



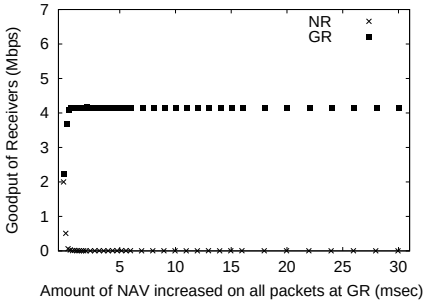
(a) *Inflated NAV in CTS*



(b) *Inflated NAV in CTS/RTS*



(c) *Inflated NAV in ACK*



(d) *Inflated NAV in RTS/CTS/Data/ACK frames*

Fig. 5. Goodput of two competing TCP flows NS-NR and GS-GR for 802.11a.

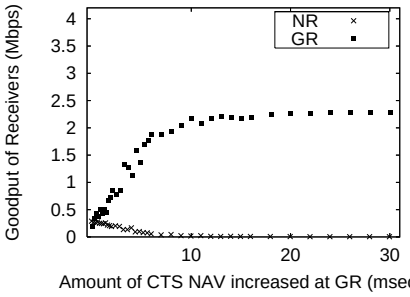


Fig. 6. Goodput of 8 TCP flows when one of the flow has a greedy receiver with an increasing CTS NAV. (802.11b)

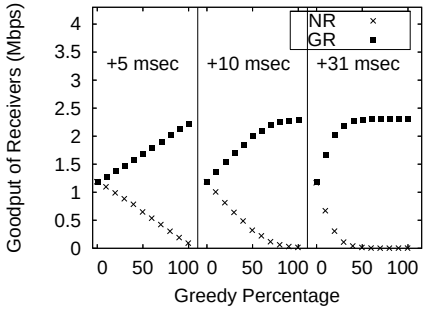


Fig. 7. Goodput of two TCP flows NS-NR and GS-GR, where GR increases NAV by 5, 10, or 31 ms, and varies greedy percentage. (802.11b)

when the receiver of one flow is greedy and inflates NAV in all of its CTS frames. We make the following observations. First, in all cases the greedy receiver obtains higher goodput than the normal receiver. Second, as we would expect, the larger increase in the greedy receiver's CTS NAV, the larger goodput gain the greedy receiver has. Moreover, with a large enough NAV value, the greedy receiver can grab the channel all the time and completely shut off the normal receiver's traffic.

Figure 4(b) shows the effect of inflating NAV on RTS and CTS frames. (A TCP receiver sends RTS frames for TCP ACK.) A very small NAV inflation can completely starve the other connection. Figure 4(c) further shows the impact of inflated ACK frames. The goodput gain from inflated ACK NAV is slightly smaller than that from inflated CTS NAV, because there are more CTS frames sent than ACK frames (ACK is sent only when RTS, CTS, and data frames are successfully received, whereas CTS is sent when RTS is received successfully). As shown in Figure 4(d), inflating NAV on all frames causes the largest damage: GS-GR pair dominates the medium even when NAV is inflated by $2ms$.

Next we evaluate the performance using 802.11a. Figure 5 summarizes the results. The high-level trend is similar: the gap between greedy and normal receivers' throughput increases with the amount of NAV inflation and the number of frames with NAV inflation. For the same amount of NAV inflation, the damage is larger in 802.11a than 802.11b because the interframe spacing and transmission time in 802.11a are smaller.

We further evaluate the effect of a greedy receiver under multiple normal sender-receiver pairs. We consider 8 flows, where one of them has a greedy receiver. As shown in Figure 6, the goodput of the greedy receiver increases with an increasing CTS NAV, at the expense of degrading the competing normal receivers. Moreover, it takes $10ms$ increase in CTS NAV for the greedy receiver to dominate the medium. In the remaining of Section V-A, unless specified otherwise, we use TCP flows since TCP is used more often.

Vary Greedy Percentage (GP): In order to make the detection difficult, a greedy receiver may not manipulate every packet it transmits. To evaluate such effect, we vary Greedy Percentage (GP), which denotes the percentage of time a greedy receiver behaves greedily. In this case, GP is the fraction of CTS frames that carry inflated NAV.

Figure 7 plots goodput of normal and greedy receivers as we vary GP and the amount of NAV inflation, and all four nodes are within communication range of each other. As we would expect, increasing GP increases the performance gain of

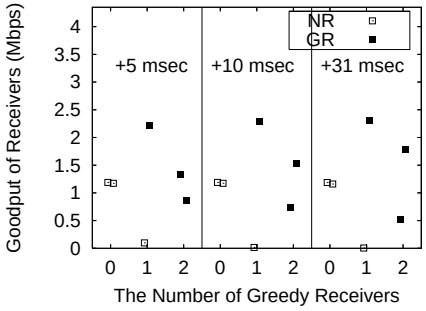


Fig. 8. Goodput under 0, 1, or 2 greedy receivers, when CTS NAV increases by 5, 10, or 31 ms. (802.11b)

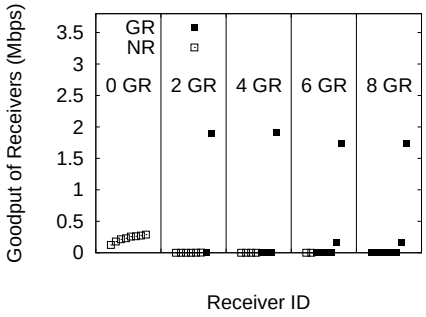


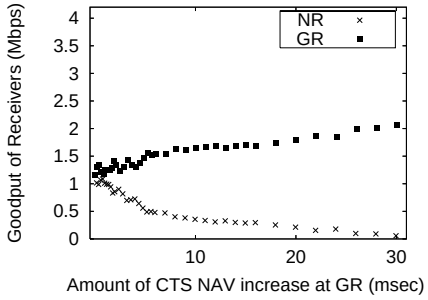
Fig. 9. Varying the number of the greedy receivers under 8 TCP flows, where all nodes can hear each other,CTS NAV is inflated 31 ms, and GP=100. (802.11b)

GR. Nevertheless even when GP is 50%, GR already receives substantially higher goodput. For example, its goodput is over 1Mbps higher than that of NR when NAV is inflated by $5ms$, and around 1.8Mbps higher when NAV is inflated by $10ms$, and completely grabs the bandwidth when NAV is inflated by $31ms$.

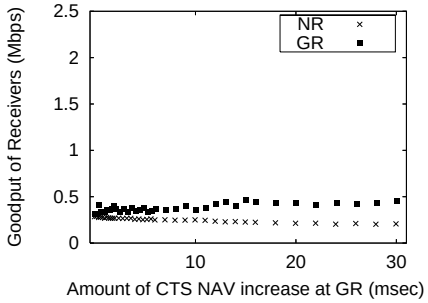
Vary the number of greedy receivers: Next we vary the number of greedy receivers. Figure 8 considers 2 sender-receiver pairs. As it shows, when both receivers are normal, they get similar goodput. When only one receiver is greedy, the greedy receiver gets significantly higher bandwidth and almost starves the normal receiver. When both receivers are greedy, their performance depends on who grabs the medium first. The one that grabs the medium earlier gets the chance to silence the other flow and has an opportunity to grab the channel again in the next round.

Figure 9 shows the results of varying the number of greedy receivers under 8 sender-receiver pairs (including normal receivers). All greedy receivers have GP=100% and increase their NAV by 31 ms. When there are more than one greedy receiver, only one greedy receiver survives and the other receivers get virtually nothing. This is because 31 ms NAV inflation is large enough so that the first one that grabs the channel reserves the medium for the subsequent transmissions. Unless there is a packet loss, the node will grab the medium for all its subsequent transmissions and starve the other flows.

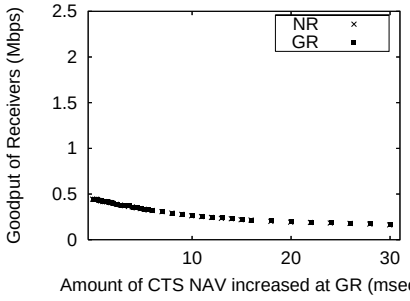
One sender with multiple receivers: So far we have studied the cases when there are as many senders as receivers. Now we examine the case where one sender sends to more than one receiver. This introduces head-of-line blocking, and reduces the damage of a greedy receiver to a certain degree. Nevertheless, even in that case, the greedy receiver can get significant gain.



(a) *2 TCP flows*



(b) *8 TCP flows*



(c) *2 UDP flows*

Fig. 10. One sender sends to multiple receivers, one of which inflates CTS NAV. (802.11b)

NAV+	1 Sender		2 Senders	
(ms)	S-NR	S-GR	NS-NR	GS-GR
0	22.281	22.388	42.177	41.775
1	19.847	24.036	46.011	42.035
2	20.961	23.590	42.424	41.005
5	16.386	27.750	30.846	47.592
10	11.723	31.345	11.810	46.272
20	10.364	36.496	4.582	49.661
31	4.519	41.715	3.203	47.452

TABLE II
AVERAGE TCP CONGESTION WINDOW.

First, we consider one sender S sending to two receivers, NR (normal receiver) and GR (greedy receiver). In this case, S does not respond to the inflated NAV from GR, since the CTS is destined to itself. Inflated NAV has the following two effects on NR. First, it prevents NR from sending CTS in response to the RTS from S in a timely manner. If the CTS is delayed long enough, the sender S assumes RTS has failed and backs off by increasing its contention window. Second, when TCP is used, an inflated NAV from GR prevents NR from sending TCP ACK in a timely manner. Figure 10(a) shows the goodput of NR and GR when both use TCP. Compared with the two-sender case, the performance gain of the greedy receiver is reduced, but the gain is still significant. Table II further compares the TCP congestion window size between 1-

BER	ACK/CTS	RTS	TCP ACK	TCP Data
$1e^{-5}$	$3.799e^{-4}$	$4.399e^{-4}$	$1.119e^{-3}$	$1.130e^{-2}$
$2e^{-4}$	$7.519e^{-3}$	$8.762e^{-3}$	$2.235e^{-2}$	$2.033e^{-1}$
$3.2e^{-4}$	$1.121e^{-2}$	$1.398e^{-2}$	$3.521e^{-2}$	$3.048e^{-1}$
$4.4e^{-4}$	$1.658e^{-2}$	$1.918e^{-2}$	$4.810e^{-2}$	$3.934e^{-1}$
$8e^{-4}$	$2.995e^{-2}$	$3.460e^{-2}$	$8.574e^{-2}$	$5.971e^{-1}$

TABLE III
BER AND THE CORRESPONDING FER

sender and 2-sender cases. As it shows, for the same NAV increase, the difference between the congestion window of normal flow and greedy flow is larger under 2-sender than that under 1-sender, but the difference is still significant.

Next we consider one sender sending to 7 normal receivers and 1 greedy receiver. Figure 10(b) plots the goodput of a greedy receiver and the average goodput of the 7 normal receivers. As we can see, there is still gain for the greedy receiver though the benefit is much smaller than competing with only one normal receiver or having multiple senders.

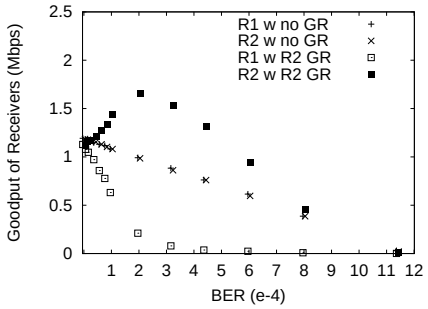
Now we consider one sender sending to a normal receiver and a greedy receiver, both using UDP. As shown in Figure 10(c), the goodput of both flows decreases with an increasing NAV, and GR receives similar goodput as NR when sharing the sender. This is because both CBR flows have the same data rate, and the queue at the sender has roughly the same number of packets to normal and greedy receivers. A larger CTS NAV from GR simply makes the sender fluctuate its contention window and increases the idle time between two transmissions, causing harm to both receivers. In comparison, under TCP the sender's queue has more packets to GR than to NR, since the TCP flow to NR slows down when NR does not send ACKs in a timely manner.

Summary: Our evaluation shows that increasing NAV is an effective greedy misbehavior. As we would expect, a larger NAV increase or a larger greedy percentage increases the gain of greedy receivers. Furthermore, the damage is larger when a greedy receiver has a separate sender from normal receivers than when the sender is shared. Finally, the impact of NAV inflation in TCP depends on which frames the greedy receiver manipulates: the impact of NAV inflation in CTS or ACK frames in TCP is smaller than that in UDP since TCP congestion control may limit the sending rate and reduces the opportunity for greedy receivers to misbehave. However, the impact on TCP traffic can further increase when the greedy receiver also modifies RTS and data frames when sending TCP ACKs.

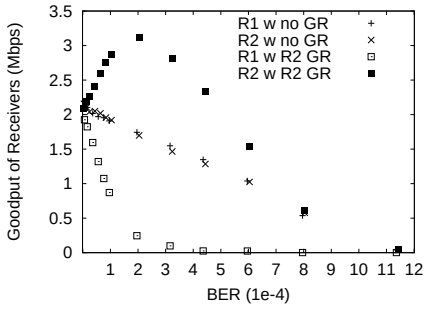
B. Misbehavior 2: Spoofing ACKs

1) TCP Traffic: We first evaluate misbehavior 2 using TCP traffic. Unless otherwise specified, we use a 4-node topology (*i.e.*, 2 senders each sending to one receiver). We place all the nodes are within communication range of each other, and the loss rates on all wireless links among all nodes are the same. We further examine the effect of the misbehavior when the connections span across both wireless and wireline links.

Vary bit error rate: First we examine the impact of a greedy receiver by varying bit error rate (BER). The greedy receiver spoofs MAC-layer ACKs for every data packet it sniffs from the sender to the normal receiver (i.e., GP=100).



(a) *802.11b*



(b) *802.11a*

Fig. 11. Goodput of two TCP flows NS-NR and GS-GR under a varying wireless link loss rate, where GR spoofs ACK on behalf of NR.

Figure 11 shows the goodput of two receivers when one of them, namely R2, misbehaves (denoted as $w\ R2\ GR$) versus when neither misbehaves (denoted as $no\ GR$) using TCP. Figure 11(a) shows 802.11b results, and Figure 11(b) shows 802.11a results. In both graphs, the x-axis shows bit error rate. The corresponding data frame error rate is shown in Table III. We make the following observations. First, when neither misbehaves, the two receivers get similar goodput. Their goodput both decreases with an increasing BER. In comparison, when one of them misbehaves, the greedy receiver gets significantly higher goodput than the normal receiver. Second, we observe that when BER is lower than $2e^{-4}$, the greedy receiver gets an increasing gain as loss rate increases. This is because an increasing loss rate means that more packets to the normal receiver have to be recovered at TCP layer after spoofing MAC-layer ACKs, thereby increasing the effectiveness of greedy misbehavior. When BER is higher than $2e^{-4}$, the greedy receiver's goodput gain gradually decreases because the number of data packets it overhears decreases, thereby decreasing the number of spoofed ACKs. Moreover, an increasing loss rate between the greedy receiver and its sender also degrades its own TCP goodput. In an extreme, when the loss rate is high enough, both TCP flows get virtually zero goodput regardless of whether one misbehaves or not. Third, comparing 802.11a and 802.11b, we observe that the general trend is similar.

Vary greedy percentage: Next we evaluate the impact of greedy percentage (i.e., how often the greedy receiver spoofs an ACK when it sniffs the other sender's data packet). Figure 12 summarizes the results. As we would expect, the goodput of greedy receiver increases as GP increases. This is true over all loss rate values. For low loss rate, the effect of spoofing is limited because most packets are correctly received at the normal receiver. For moderate loss rate, a

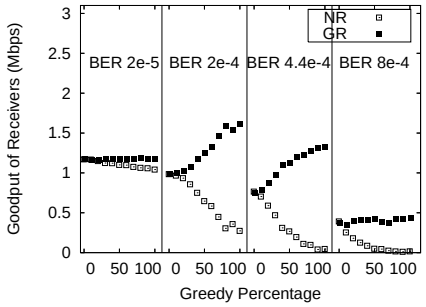


Fig. 12. Goodput of two TCP flows NS-NR and GS-GR, when greedy percentage and loss rates vary. (802.11b)

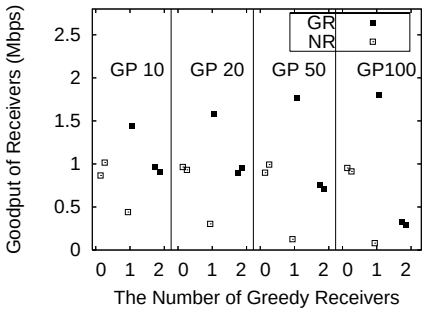


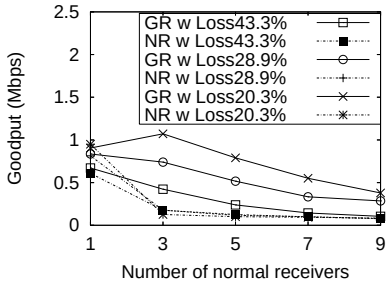
Fig. 13. Goodput under 0, 1, or 2 greedy receivers (All flows use TCP, and BER=$2e^{-4}$). (802.11b)

significant number of packets are lost at the normal receiver, making spoofing ACK an effective attack. For high loss rate, spoofing ACK continues to allow the greedy receiver to get more goodput than the normal receiver, even though the greedy receiver also suffers degradation due to its high loss rate.

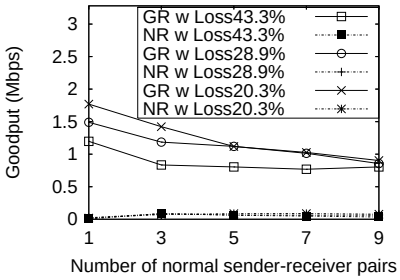
Vary the number of greedy receivers: We further evaluate the performance of 2 TCP flows under 0, 1, or 2 greedy receivers. As shown in Figure 13, the total goodput decreases when both receivers misbehave. This is because in this case both receivers spoof the other's MAC-layer ACK, which effectively disables MAC-layer retransmission and makes the loss propagated to TCP layer. A larger GP causes MAC-layer retransmission to be disabled more often, and results in larger reduction in goodput.

Vary the number of sender-receiver pairs: Next we consider one greedy receiver competes with a varying number of normal receivers. Figure 14(a) compares the average goodput of a greedy receiver and normal receivers when they share one AP, and Figure 14(b) shows the results when each receiver receive data from a different AP with 802.11b. As they show, in both cases the average throughput of greedy receiver is higher than that of the normal receivers. The difference in goodput between the greedy and normal receivers is smaller under one AP due to head-of-line blocking.

TCP sender at remote site: So far we consider the connections span only wireless links. Next we consider the case where the two connections span both wireless and wireline links, as shown in Figure 15(a). We vary the wired link latency from $2ms$ to $400ms$, and set BER of both wireless links to $2e^{-5}$. Figure 15 compares goodput under no greedy receiver (denoted as $no\ GR$) versus under one greedy receiver $R2$ (denoted as $w\ \ R2\ GR$). We observe that increasing wireline latency initially increases the gap between the normal



(a) *one AP*



(b) *multiple APs*

Fig. 14. One greedy receiver competes with a varying number of NS-NR pairs. (TCP, 802.11b)

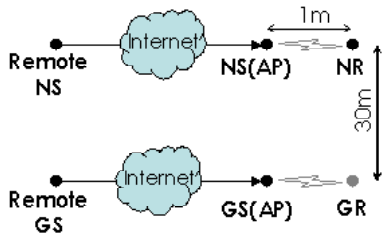
and greedy receiver. This is because an increasing wireline latency makes end-to-end loss recovery more expensive. When the wireline latency is beyond $200ms$, the goodput of greedy receiver starts to decrease, even though it still significantly out-performs the normal receiver. This is because TCP ACK-clocking reduces the greedy receiver's goodput as its delay increases, and the goodput gain from the normal receiver is not enough to offset such drop.

Figure 16 further shows the result when we vary the greedy percentage (GP) under five different values of wireline latency. As we would expect, increasing GP enlarges the performance gap between the greedy and normal receivers. In addition, we observe the performance gain of the greedy receiver is largest when the wireline latency is around 200 ms – only spoofing 20 % of DATA frames it sniffs, GR achieves 51.78% more goodput of than its normal value, causing NR to perform 63.36% worse than its normal value.

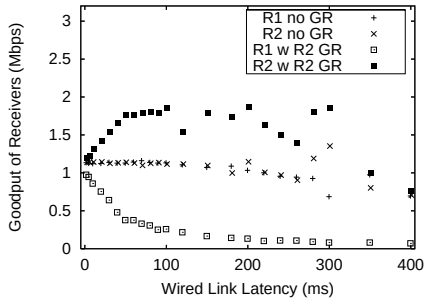
2) UDP Traffic: Now we evaluate the spoofing ACK using UDP traffic. We consider a 3-node topology (*i.e.*, 1 AP sending to two receivers). Figure 17 summarizes the results using the same notation as above. As we would expect, when neither misbehaves, the two receivers get similar goodput. When one of them misbehaves, GR gets higher throughput than NR. The performance gain of GR in UDP is less pronounced as in TCP, because disabling MAC-layer retransmissions interacts with TCP congestion control and causes larger damage.

C. Misbehavior 3: Sending Fake ACKs

For misbehavior 3, a greedy receiver sends an ACK even upon receiving a corrupted data frame. As mentioned in Section IV-C, this misbehavior is effective when the greedy receiver uses UDP and generates additional traffic to the MAC-layer to compensate for the reduced service time caused by disabling MAC-layer retransmission. So our evaluation uses 4-node topologies, where two APs, $S1$ and $S2$, each send traffic



(a) *Simulation topology*



(b) *Simulation results*

Fig. 15. Goodput under remote TCP senders, where both wireless links to the greedy and normal receiver have BER=$2e^{-5}$. (802.11b)

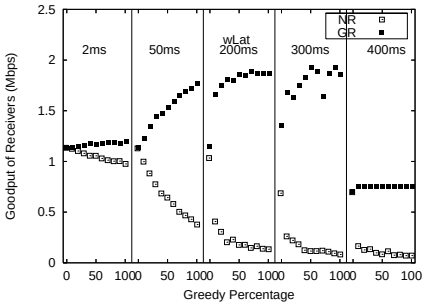
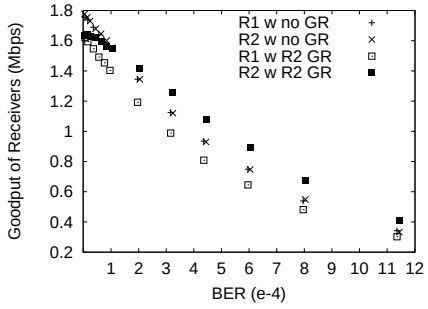
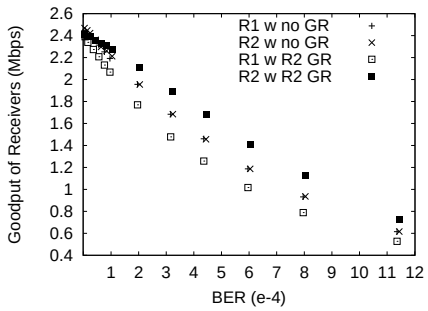


Fig. 16. Varying greedy percentage and wireline link latency, where both wireless links to the greedy and normal receiver have BER = $2e^{-5}$. (802.11b)

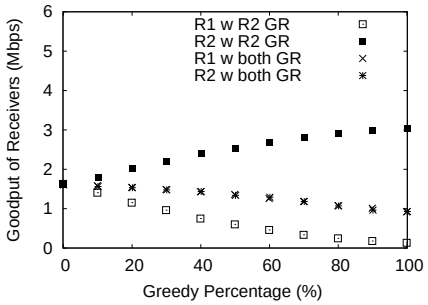


(a) *802.11b*

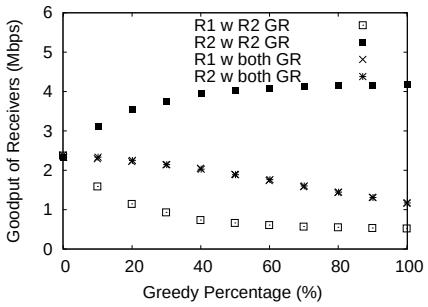


(b) *802.11a*

Fig. 17. Goodput of two UDP flows S-NR and S-GR under a varying wireless link loss rate, where GR spoofs ACK on behalf of NR.



(a) 802.11b



(b) 802.11a

Fig. 18. Goodput of two UDP flows $S1$-$R1$ and $S2$-$R2$, when $S1$ and $S2$ are hidden terminals.

	No GR		1 GR		2 GRs	
	$NR1$	$NR2$	NR	GR	$GR1$	$GR2$
802.11b	124.35	125.69	362	43.31	77	76.34
802.11a	92.55	97.02	220.52	28.91	47.52	47.33

TABLE IV

CONTENTION WINDOW SIZE OF NR AND GR WITH GP 100% WHEN BOTH SENDERS ARE HIDDEN TERMINALS USING UDP

to its receiver $R1$ and $R2$, respectively, as fast as possible. We create data loss using one of the following two ways. We disable RTS-CTS exchange and place two receivers next to each other and senders far apart from each other to create the hidden terminal problem. Alternatively, we create loss by injecting random loss of bit-error-rate (BER) of $2e^{-5}$ when the two sender-receiver pairs are within communication range of each other. In both cases, the two flows experience similar loss rates.

With collision losses of hidden terminals: Figure 18 compares the goodput of two UDP flows, $S1 - R1$ and $S2 - R2$, when only $R2$ misbehaves, and when both $R1$ and $R2$ misbehaves under varying greedy percentage. In this scenario, $S1$ and $S2$ are hidden terminals thus $R1$ and $R2$ experience collision losses. We can observe the following. First, when only $R2$ misbehaves, an increasing greedy percentage increases the discrepancy between the goodput of the normal receiver $R1$ and greedy receiver $R2$. When GP=100% (i.e., greedy receiver fakes ACK on every corrupted data packet), the greedy receiver significantly outperforms the other connection. This is because faking ACKs makes GS's contention window (CW) considerably smaller than NS's CW, as shown in Table IV. Second, when both receivers are greedy, they both suffer. This is because faking ACK essentially disables exponential backoff in 802.11 and lets senders send faster than they should, creating more collisions. Thus, under traffic-

Data error rate	no GR (Mbps)		1 GR (Mbps)		2 GRs (Mbps)	
	$R1$	$R2$	$R1$	$R2(GR)$	$R1(GR)$	$R2(GR)$
0.2	1.44	1.43	1.2	1.43	1.47	1.47
0.5	0.92	0.91	0.59	2.49	1.03	1.03
0.8	0.63	0.63	0.32	1.11	0.71	0.75

TABLE V

GOODPUT OF TWO UDP FLOWS UNDER INHERENT WIRELESS LOSSES UNDER 802.11B

induced losses arising from hidden terminals, sending fake ACK can be harmful to greedy receivers when there is no normal receiver.

With inherent wireless medium losses: Next we compare the performance of the two UDP flows, $S1 - R1$ and $S2 - R2$ with inherent wireless medium losses. When the loss is inherent wireless medium losses (e.g., low received signal strength), faking ACKs improve the goodput by 2-12% when data frame loss rate varies from 0.2 to 0.8, as shown in Table V. In this case, unlike traffic-induced loss case, performing exponential back-off does not help reduce losses and only unnecessarily reduces the sending rate. Faking ACKs avoids such unnecessary rate reduction and improves performance.

Different loss rates on the two flows: Our next evaluation is to understand whether a greedy receiver's performance gain under packet losses is no more than a normal receiver when its link is loss free. So we inject random loss to only one flow, and let both receivers behave normally. When both flows have BER of $5e^{-4}$, the greedy and normal receivers obtain 2.61 Mbps and 1.086 Mbps, respectively. In comparison, when both receivers are normal, one flow with BER of $5e^{-4}$, and the other with no loss, the one with no loss has 2.64Mbps and the other has 1.096 Mbps. So effectively the greedy receiver pretends to be a normal receiver without packet losses. Under inherent wireless medium loss, faking ACKs can be considered as a useful surviving technique. However, this is not recommended under traffic-induced losses.

Vary the number of sender-receiver pairs: Finally we consider multiple APs each transmit to one client as fast as possible, where one of the clients is a greedy receiver and all of the clients experience the same loss rates. Figure 19 shows the goodput as we vary the number of sender-receiver pairs. We observe that the impact of greedy receiver increases with the loss rate, because a higher loss rate means more opportunities for the greedy receiver to fake ACKs. Moreover, the absolute difference in the goodput of greedy and normal receivers decreases as the number of normal receivers increases due to a decreasing per-flow goodput. Interestingly, their relative difference in goodput remains high for all the numbers of receivers considered.

VI. EVALUATION IN TESTBED

In this section, we evaluate the performance impact of greedy receivers in a testbed consisting of 4 DELL Dimensions 1100 PCs (2.66 GHz Intel Celeron D Processor 330 with 512 MB of memory). For the first two misbehaviors, we use two senders, each sending to one receiver. For the third misbehavior, spoofing ACK, we use one sender transmitting to two receivers. In all cases, the senders send as fast as possible. The nodes are all at fixed locations within communication range of one another. Each node runs Fedora Core 4 Linux, and is equipped with 802.11 a/b/g NetGear WAG511 using MadWiFi. We use 802.11a to avoid interference with campus 802.11b wireless LAN in the building. We enable RTS/CTS and use a fixed rate of 6Mbps.

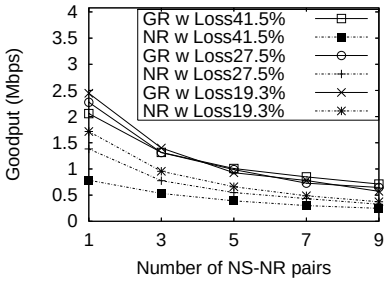


Fig. 19. One greedy receiver competes with a varying number of NS-NR pairs.

	no GR (Mbps)		1 GR (Mbps)	
	$NR1$	$NR2$	GR	NR
Inflate NAV on RTS of TCP Ack	2.28	2.51	4.41	0.04

TABLE VI

TCP THROUGHPUT WHEN GR INFLATES NAV OF RTS FOR TCP ACKS.

Inflating NAV: As a receiver of TCP traffic, GR can inflate NAV in RTS and DATA frames when it sends TCP ACKs, since TCP ACKs are considered as DATA frame to MAC-layer. In MadWifi, we can easily modify the NAV in RTS frames. Table VI shows the NAV inflation result using TCP where GR inflates NAV in RTS frames for TCP ACKs to the maximum value, $32767\mu s$. As it shows, without greedy receiver, $NR1$ and $NR2$ shares the medium fairly, but when $NR1$ becomes greedy receiver, it consumes almost all the bandwidth and the normal receiver only receives 0.04 Mbps.

Next we consider UDP flows. In this case, a greedy receiver can only inflate CTS and ACK. In Madwifi, CTS and ACK frames are sent automatically in hardware and we cannot modify NAV in these frames. Instead, we can disable the automatic ACK and CTS by changing Atheros configuration register number 0x8048 to 0x7, and inject our own ACK and CTS with inflated NAV. The packet injection can be implemented either inside the MadWiFi driver or as an application layer program. For simplicity, we adopt the latter approach to inject ACK and CTS packets with inflated NAV value of $32767\mu s$ via the raw interface. A CTS frame with inflated NAV is injected upon the receipt of an RTS frame, and an ACK frame is injected upon the receipt of a data frame. However, since we implement in an application layer program, the turn around time for sending ACK and CTS frames is around $1\ ms$, which is larger than their timeout values. Therefore we do not disable automatic ACK and CTS, but run our packet injection code in parallel. (If we implement packet injection in MadWifi, the turn-around time is $50\ \mu s$, within the maximum ACK/CTS timeout of $409\ \mu s$, so we can disable automatic ACK/CTS.) For the purpose of NAV inflation, the larger turn around time is not an issue since the slightly delayed frames with inflated NAV still achieve the goal of silencing nearby nodes.

Table VII compares the UDP throughput of the senders

	no GR (Mbps)		1 GR (Mbps)	
	$NR1$	$NR2$	GR	NR
no RTS/CTS inflated NAV on ACK	2.73	2.85	4.94	0.08
with RTS/CTS inflated NAV on CTS	2.47	2.67	4.65	0.08
with RTS/CTS inflated NAV CTS/ACK	2.47	2.67	4.65	0.05

TABLE VII

UDP THROUGHPUT WHEN GR INFLATES NAV IN TESTBED.

	no GR (Mbps)		1 GR (Mbps)	
	$NR1$	$NR2$	GR	NR
without RTS/CTS	2.68	1.96	3.51	0.98

TABLE VIII

TESTBED EMULATION OF SPOOFING ACK USING TCP.

when there is no greedy receivers versus when one receiver is greedy. The reported goodput is the median over 5 runs, where each run lasts 30 seconds. As it shows, without greedy receiver, both senders achieves similar throughput. When one receiver is greedy, the greedy receiver's flow gets virtually all the bandwidth while starving the normal receiver's flow.

Emulating ACK spoofing: We study the impact of spoofing ACK by having one sender transmit to two receivers using TCP. To emulate the effect of ACK spoofing, we modify the sender to disable MAC-layer retransmissions when it transmits to the normal receiver, while perform MAC-layer retransmissions as usual when it transmits to the greedy receiver. As shown in Table VIII, when greedy receiver successfully spoofs an ACK on behave of the normal receiver, its goodput increases by 30% while the normal receiver's goodput decreases by half.

Emulating fake ACKs: Finally we examine the impact of sending fake ACK by letting a sender send UDP traffic to two receivers. We emulate the effect of sending fake ACKs by setting the sender's maximum contention window to minimum contention window when it transmits to the greedy receiver. As we can see from Table IX, this misbehavior enables the greedy receiver to grab the medium faster than the competing flow and obtaining higher throughput.

VII. DETECTING GREEDY RECEIVERS

In this section, we present techniques to detect and mitigate greedy receiver misbehaviors. We assume that senders are well-behaving and do not collude with greedy receivers. Fig. 20 shows a flow-chart of our countermeasure scheme. The scheme can be implemented at any node in the network, including APs and clients. The more nodes implementing the detection scheme, the higher likelihood of detection. Next we describe how to detect inflated NAV, spoofed ACKs, and fake ACKs.

A. Detecting Inflated NAV

Inflated NAV affects two sets of nodes: (i) those within communication range of the sender and receiver, and (ii) those

	no GR (Mbps)		1 GR (Mbps)	
	$NR1$	$NR2$	GR	NR
without RTS/CTS	2.08	2.99	2.79	2.35

TABLE IX

TESTBED EMULATION OF SENDING FAKE ACK USING UDP.

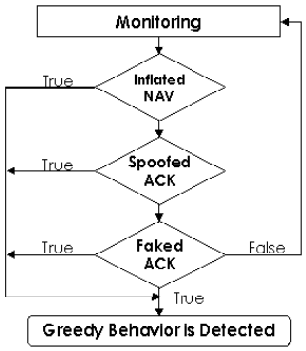


Fig. 20. Detecting greedy receivers.

outside the communication range of the sender but within communication range of the receiver. The first set of nodes know the correct NAV, since they overhear the sender's frame and can directly compute the correct NAV from the receiver by subtracting the duration of sender's frame. Therefore these nodes can directly detect and correct inflated NAV. The second set of nodes can infer an upperbound on a receiver's NAV using the maximum data frame size (*e.g.*, 1500 bytes, Ethernet MTU). For Giga Ethernet, the jumbo frame MTU is 9000 bytes. However, since we focus on the traffic going to and from the Internet, and based on extensive Internet traffic measurement study [18] the packet size on the Internet is within 1500-bytes. Therefore it is reasonable to assume MTU of 1500 bytes.

If the NAV in CTS or ACK exceeds the expected NAV value, greedy receiver is detected. (In fact, without fragmentation, NAV in ACK should always be 0.) We can further locate the greedy receiver using received signal strength measurement from it. To recover from this misbehavior, nodes will ignore the inflated NAV and replace it with the expected NAV to use for virtual carrier sense.

B. Detecting Spoofed ACKs

To detect greedy receivers that spoof ACKs on behalf of normal receivers, we use their received signal strength. More specifically, let RSS_N denote the received signal strength from the original receiver, RSS_C denote the received signal strength in the current ACK frame, and $Thresh_{cap}$ denote the capture threshold. RSS_N can be obtained using a TCP ACK from that receiver, assuming TCP ACK is not spoofed If RSS_C is significantly different from RSS_N, the sender reports greedy misbehavior. Furthermore, when $\frac{RSS_N}{RSS_C} \geq Thresh_{cap}$, the sender can directly recover from this misbehavior by ignoring the received ACK. This is because in this case the original receiver must have not received data and sent ACK, otherwise the ACK coming from the original receiver would have captured the spoofed ACK; ignoring such MAC-layer ACKs allow the sender to retransmit the data at the MAC-layer as it should.

To examine the feasibility of using RSS measurements for detecting spoofed ACKs, we collect Received Signal Strength Indication (RSSI) measurements from our testbed, consisting of 16 nodes spread over one floor of an office building. For each run, one sender sends UDP broadcast packets and

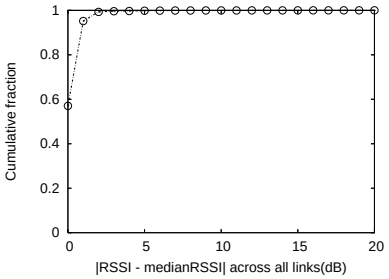


Fig. 21. CDF of $|RSSI - medianRSSI|$ over all links.

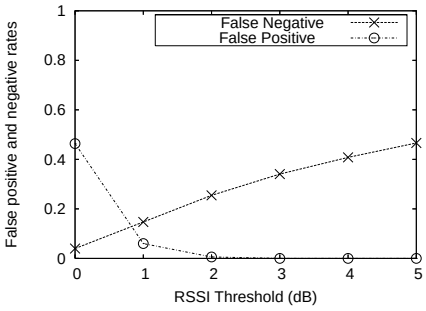


Fig. 22. False positive and false negative vs. RSSI threshold.

all other nodes record RSSI values for each packet using Click [4]. (RSSI is defined as $10\log_{10}\frac{S+I}{N}$, where S denotes received signal strength, I denotes interference, and N denotes thermal noise [16].) As shown in Figure 21, around 95% RSSI measurements differ from median RSSI of that link within 1 dB. This suggests that RSSI does not change much during a short time interval, and we can use large change in RSSI to identify spoofed ACKs.

Based on the above observation, a sender determines a spoofed ACK if $|RSSI_{median} - RSSI_{curr}| > RSSIThresh$, where $RSSI_{median}$ is the median RSSI from the true receiver, $RSSI_{curr}$ is the RSSI of the current frame, and $RSSIThresh$ is the threshold. The accuracy of detection depends on the value of $RSSIThresh$. Fig. 22 plots the false positive and false negative rates as $RSSIThresh$ varies from 0 to $5dB$, where false positive is how often the sender determines it is a spoofed ACK but in fact it is not, and false negative is how often the sender determines it is not a spoofed ACK but in fact it is. As it shows, using 1 dB as the threshold achieves both low false positive and low false negative rates.

The previous detection is effective when RSSI from NR is relatively stable and RSSI from GR is different from NR. To handle highly mobile clients, which experience large variation in RSSI, the sender can use a cross-layer approach to detect the greedy behavior. For each TCP flow, it maintains a list of recently received MAC-layer ACK and TCP ACK. Greedy receiver is detected when TCP often retransmits the packet for which MAC-layer ACK has been received. This detection assumes wireline loss rate is much smaller than wireless loss rate, which is generally the case.

C. Detecting Faked ACKs

To detect greedy receivers that send MAC-layer ACKs even for corrupted frame, the sender compares the MAC-layer loss with the application layer loss rate. The latter can be estimated using active probing (e.g., ping). Since packets are corrupted, GR cannot send ping response and we can measure the true application loss rate. If loss rate is mainly from wireless link, $applicationLoss \approx MACLoss^{maxRetries+1}$, when packet losses are independent. If $applicationLoss > MACLoss^{maxRetries+1} + threshold$, the sender detects faked ACKs, where $threshold$ is used to tolerate loss rate on wireline links when the connection spans both wireless and wireline. The appropriate value of threshold depends on the loss rate on the wireline links.

VIII. Evaluation of Detection

We implement in NS-2 the greedy receiver countermeasure (GRC) against inflated NAV and ACK spoofing described in Section VII.

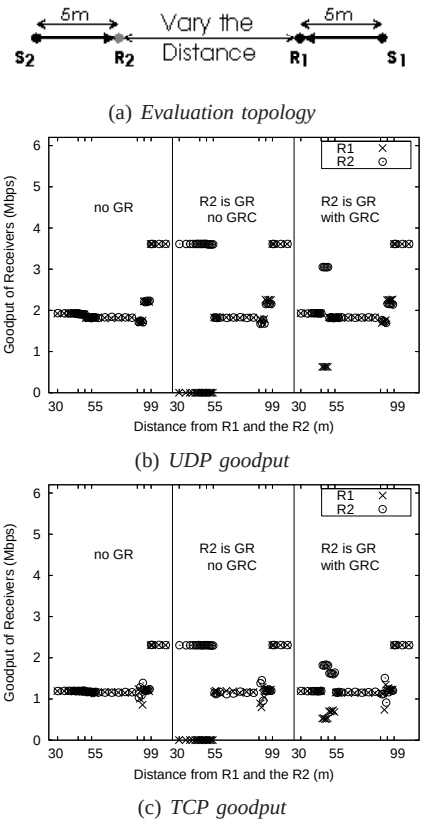


Fig. 23. GRC effectively detects and mitigates inflated CTS NAV.

First we evaluate the countermeasure against inflated CTS NAV using the the topology in Figure 23(a), where communication and interference ranges are 55m and 99m, respectively. Figure 23(b) compares the UDP performance under the following three cases (from left to right) : (i) no greedy receiver, (ii) one greedy receiver with no GRC, and (iii) one greedy receiver and with GRC. As we can see, without a greedy receiver, the two flows get similar goodput. The goodput jumps around $99m$, because the two senders do not interfere beyond this distance. When $R2$ is greedy, $R2$ dominates the medium and completely shuts off $R1$ when all four nodes are within communication range. Beyond $55m$, $R2$'s inflated CTS NAV cannot be heard by $R1$ and $S1$, so the goodput of the two flows are similar beyond $55m$. So inflated CTS NAV is effective only

when distance is below $55m$, and we focus on this region. We observe that GRC effectively detects and mitigates the inflated NAV. In particular, the goodput of the two flows are similar when distance is below 45 m, since $S1$ and $R1$ both hear $S2$'s RTS and know the true packet size. As the distance further increases, NS does not hear RTS from GS and has to assume the maximum packet size 1500 bytes, which is 46.48% larger than the actual data packet size. In this case, $R2$ receives higher goodput. Nevertheless, compared with no GRC, the normal receiver no longer starves. Similar trends are observed under TCP traffic, as shown in Figure 23(c).

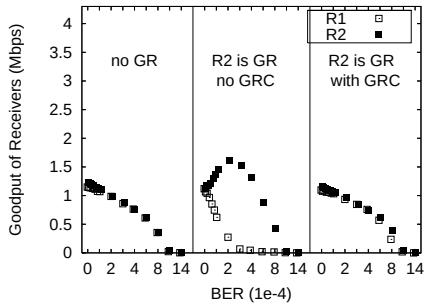


Fig. 24. GRC effectively detects and recovers from ACK spoofing under varying BER.

Next we consider a greedy receiver that spoofs ACKs. We compare the goodput of two competing flows under a varying loss rate, where the loss rates on the two flows are the same and losses are both randomly generated. As shown in Figure 24, without a greedy receiver, the goodput of the two flows are similar: both gradually decrease as BER increases from 0 to $14e^{-4}$. When $R2$ is greedy, its flow dominates the medium and degrades $R1$'s performance when no GRC is used. With GRC, both flows fairly share the medium: their goodput closely follow the goodput curves under no greedy receiver. This demonstrates the effectiveness of the GRC.

IX. Conclusion

As the popularity of hotspot networks continues to grow, it is increasingly important to understand potential misuses and guard against them. In this paper, we identify a range of greedy receiver misbehaviors, and evaluate their effects using both simulation and testbed experiments. Our results show that greedy receiver misbehavior can cause serious degradation in other traffic, including starvation. We further develop techniques to detect and mitigate the misbehaviors and demonstrate their effectiveness.

This paper focuses on the effects of greedy receivers in fixed rate environments. Rate adaptation introduces strong interactions with several misbehaviors. In particular, the damage of faking ACKs (*i.e.*, misbehavior 3) may reduce under auto-rate, since without correct feedback the transmitter may not choose the best modulation scheme and cause performance degradation. In contrast, the damage of spoofing ACKs (*i.e.*, misbehavior 2) can increase with auto-rate. With ACK spoofing, the sender may not be able to select a good data rate to use and incur significant performance degradation, which may benefit the greedy receiver. We plan to explore the effects of auto-rate in depth as part of our future work.

References

[1] I. Aad, J. Hubaux, and E. Knightly. Impact of denial of service attacks on ad hoc networks. *IEEE Transactions on Networking (ToN)*, 2007.

[2] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proc. of 12th USENIX Security Symposium*, Aug. 2003.

[3] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On selfish behavior in CSMA/CA networks. In *Proc. of IEEE Infocom*, Mar. 2005.

[4] Click. http://pdos.csail.mit.edu/click/.

[5] I. C. S. L. M. S. Committee. *IEEE 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications*, Aug. 1999.

[6] L. Guang, C. Assi, and A. Benslimane. Modeling and analysis of predictable random backoff in selfish environments. In *ACM/IEEE MSWiM*, 2006.

[7] Y. Hu and A. Perrig. A survey of secure wireless ad hoc routing. *IEEE Security & Privacy, special issue on Making Wireless Work*, pages 28–39, 2004.

[8] In-stat. `http://www.in-stat.com/catalog/Wcatalogue.asp?id=167`.

[9] Revenue from wireless hotspots. `http://blogs.zdnet.com/ITFacts/index.php?blogthis=1&p=9319`.

[10] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *Proc. of ACM MOBICOM*, Sept. 2002.

[11] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants). In *Proc. of ACM SIGCOMM*, Aug. 2003.

[12] P. Kyasanur and N. Vaidya. Detection and handling of MAC layer misbehavior in wireless networks. In *IEEE Transactions on Mobile Computing*, Apr. 2004.

[13] The network simulator – ns-2. http://www.isi.edu/nsnam/ns/.

[14] M. Raya, I. Aad, J. P. Hubaux, and A. E. Fawal. Domino: Detecting mac layer greedy behavior in ieee 802.11 hotspots. *IEEE Transactions on Mobile Computing*, 2006.

[15] M. Raya, J. P. Hubaux, and I. Aad. DOMINO: A system to detect greedy behavior in IEEE 802.11 hotspots. In *Proc. of MobiSys*, Sept. 2004.

[16] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Z. an. Measurement-based models of delivery and interference. In *Proc. of ACM SIGCOMM*, 2006.

[17] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP congestion control with a misbehaving receiver. *ACM Computer Communications Review*, Oct. 1999.

[18] R. Sinha, C. Papadopoulos, and J. Heidemann. Internet packet size distributions: Some observations. *Technical Report ISI-TR-2007-643, USC/Information Sciences Institute*, May 2007.

[19] D. Tang. Analysis of a local-area wireless network. In *Proc. of MOBICOM*, Sept. 2000.

[20] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proc. of ACM MobiHoc*, May 2005.

Mi Kyung Han is a Ph.D. candidate in the Department of Computer Sciences at the University of Texas at Austin. She received her Bachelor of Computer Science from Korea Advanced Institute of Science and Technology in 2005. Her research area includes wireless networks, network protocol design, and network management.

Lili Qiu received her Ph.D. from Cornell University in 2001. She is currently an Assistant Professor in the Department of Computer Sciences at the University of Texas at Austin. Her research area is computer networks, with special focuses on wireless network management, content distribution, Internet measurement, and overlay networks. She is a recipient of NSF career award (2006), and a senior IEEE member. She was a researcher at Microsoft Research during 2001-2004.