Greedy Receivers in IEEE 802.11 Hotspots

Mi Kyung Han, Brian Overstreet, Lili Qiu
Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712
{hanmi2,overstre,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: Greedy receiver, wireless LAN.

1 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 [5, 6], 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 [1, 9, 11]. 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 [7, 13]. 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. 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 in Section 2, and survey related work in Section 3. We present a range of greedy receiver misbehaviors in Section 4. We quantify their damage using simulation and testbed experiments in Section 5 and Section 6, respectively. We describe techniques to detect and mitigate greedy receiver misbehavior in Section 7, and evaluate its effectiveness in Section 8. We conclude in Section 9.

2 Background of IEEE 802.11

In IEEE 802.11 DCF [3], before transmission, a station first checks to see if the medium is available by 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. 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}.

3 Related Work

The serious damage caused by misbehaving MAC has received increasing attention in wireless research community. For example, Bellardo and Savage [1] studied denial of service attacks in 802.11. Kyasanur and Vaidya [9] identified that selfish senders can get significantly more bandwidth than regular senders by modifying the backoff value in IEEE 802.11. Raya et al. [11] developed DOMINO, a software installed on access points to detect and identify greedy stations. More recently, Cagalj et al. [2] 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 [14], routing attacks [4], and selfish TCP behavior and attacks [8, 12]. In particular, our work is inspired by [12], 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 [12], we study receiver misbehavior at MAC-layer.

4 Greedy Receiver

In this section, we present three types of greedy receiver misbehaviors. 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.

4.1 Misbehavior 1: Increasing NAV

IEEE 802.11 uses NAV to perform virtual carrier sensing. Greedy receivers can increase their goodput (i.e., received rate) by increasing NAV. Our work complements the previous studies on NAV inflation [1, 11] in the following aspects. First, we focus on greedy receiver misbehavior, whereas [11] focuses on greedy senders and [1] 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 [1]. Second, we present a simple analysis to model the effect of NAV inflation in Section 5. Third, we will use extensive evaluation to study the effect of NAV inflation in various scenarios.

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 [3],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.

4.2 Misbehavior 2: Spoofing ACKs

Upon a packet loss, a TCP sender reduces its sending rate by decreasing its congestion window (*i.e.*, the maximum amount of unacknowledged data allowed by the TCP sender). 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 A spoofed ACK has two effects. First, when the original receiver (NR) does not receive the data frame, 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 5.

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 [14]. Therefore Section 5, we focus on the first effect – disabling MAC-layer retransmissions.

To study the first effect, we consider capture effects: When the two packets are received simultaneously, if their received signal strength ratio 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. (The performance degradation caused by greedy receiver would be even larger under both jamming and disabled MAC retransmissions.)

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

Table 1. Testbed measurement shows that most corrupted packets preserve source and destination MAC addresses.

4.3 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 packets).

Applicable Scenarios This misbehavior is effective under the following two conditions. First, the traffic to GR is carried by non-TCP connections (to avoid interacting with TCP congestion control). Second, the link from GS to GR is lossy.

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

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 1 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 practical attack since most of corrupted packets preserve MAC addresses.

Effects GR sending ACK in the presence of corrupted data effectively prevents GS from doing exponential backoff, which increases GR's 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 ACK is transmitted under corrupted/lost packets. However they differ in that misbehavior 2 targets TCP traffic by exploiting its rate reduction upon packet losses whereas misbehavior 3 targets non-TCP traffic by avoiding MAC-layer exponential backoff under packet losses.

5 Evaluation of Greedy Receivers in Simulation

In this section, we use Network Simulator 2 (NS2) [10] to quantify the damage caused by greedy receivers. We use 802.11b for all simulation evaluation, since 802.11b is commonly used in hotspot networks. All the senders behave normally, which correspond to APs that are under the control of hotspot providers and do not misbehave. We consider NS sends to NR, and GS sends to GR, where GR denotes a greedy receiver, and NS, GS and NR all behave normally. Our evaluation uses both TCP and UDP, both of which use 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, which is the received data rate of uncorrupted packets. 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.

5.1 Misbehavior 1: Increasing NAV

We randomly place nodes so that all of them can hear each other. 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 greedy 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 corresponding to the TCP ACK.

Vary the amount of NAV inflation: We vary the value of NAV used by greedy receivers by changing α in $n + \alpha \cdot 100$, where n is the original NAV value before inflation, and 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. Fig. 1 shows the goodput of a normal receiver and a greedy receiver, competing with each other and both using UDP. The greedy receiver

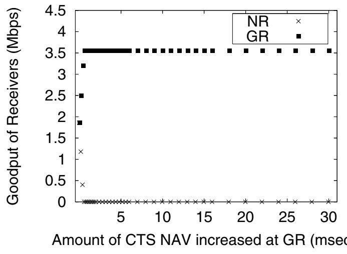


Figure 1. Goodput of two UDP flows NS-NR and GS-GR, where GR inflates CTS NAV.

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. 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 $Pr[B_{GS} \leq B_{NS} + v + 1]$. v is added to B_{NS} because GS starts count-down 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 timeslot. 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}. We find that as NAV increases NS's average CW increases due to increasing collisions, whereas GS's average CW stays close to CW_{min} because the fraction of collided packets does not change much due to an increasing number of packets sent by GS. Based on the above observation, we have the following relationship:

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

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

We evaluate the accuracy of our model by plugging the distributions of CW into Equation 1 and 2. Fig. 2

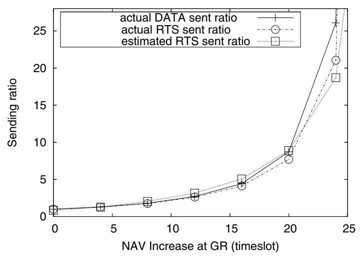
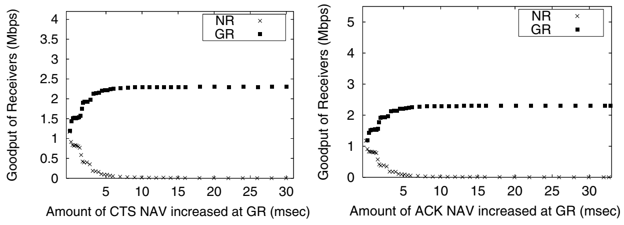
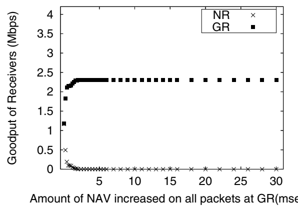


Figure 2. Sending ratio between two competing UDP flows $GS - GR$ vs. $NS - NR$.



(a) Inflated NAV in CTS (b) Inflated NAV in ACK



(c) Inflated NAV in RTS/CTS/Data/ACK frames

Figure 3. Goodput of two competing TCP flows NS-NR and GS-GR.

compares the estimated and actual RTS sent ratios from GS and NS. As we can see, our model accurately estimates the RTS sent ratio, which is very close to the actual data sent ratio. The actual RTS and data sent ratios are slightly different because of packet losses.

TCP traffic: Fig. 3(a) shows the goodput of two competing TCP flows, 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.

Fig. 3(b) further shows the effect of inflating NAV on 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 we would expect, inflating NAV on all frames causes the largest damage: GS-GR pair dominates the medium even when NAV is inflated by $2ms$, as shown in Fig. 3(c).

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.

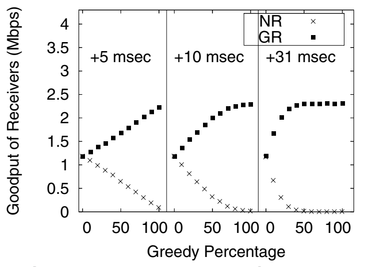


Figure 4. Goodput of two TCP flows NS-NR and GS-GR, where GR increases NAV by 5, 10, or 31 ms, and varies greedy percentage.

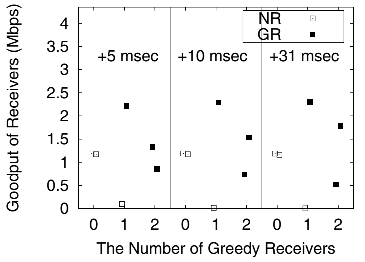


Figure 5. Goodput under 0, 1, or 2 greedy receivers, when CTS NAV increases by 5, 10, or 31 ms.

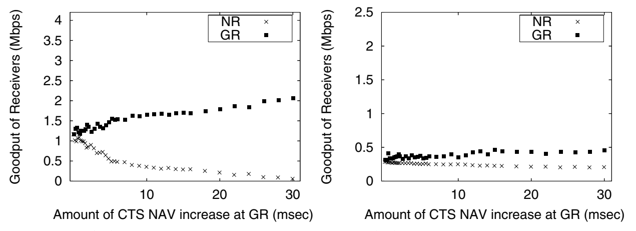
We observe that the goodput of the greedy receiver increases with an increasing CTS NAV, at the expense of degrading the goodput of 7 competing normal receivers. Moreover, it takes $10ms$ increase in CTS NAV for the greedy receiver to dominate the medium. In the remaining of Section 5.1, unless specified 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 carrying inflated NAV.)

Fig. 4 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, GR has a larger gain with an increasing GP. 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. Fig. 5 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.

We further study the case of 8 sender-receiver pairs,



(a) 2 TCP flows (b) 8 TCP flows

Figure 6. One sender sends to multiple receivers, one of which inflates CTS NAV.

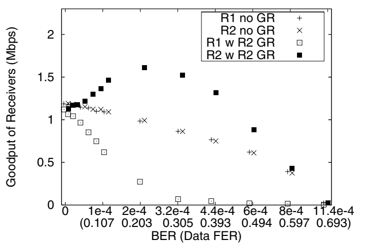


Figure 7. Goodput of two TCP flows NS-NR and GS-GR under a varying wireless link loss rate. The x-axis shows bit error rate on the top and data frame error rate on the bottom.

where all greedy receivers have GP=100% and their NAV is increased by 31 ms. When there are more than two greedy receivers, only two of the greedy receivers survive and the rest of the receivers have 0 goodput. This is because when the amount of NAV increase is large enough, the first few that grab the channel will dominate the medium.

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.

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. Fig. 6(a) shows the goodput of NR and GR. Compared with the case of two sender-receiver pairs, the goodput increase for the greedy receiver is reduced, even though the gain is still significant. Next we consider one sender sending to 7 normal receivers and 1 greedy receiver. Fig. 6(b) shows that 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.

Next we consider one sender sending to a normal receiver and a greedy receiver using UDP. (Figure is omitted for brevity.) 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 without changing splitting ratio between the two 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; however the impact on TCP traffic can further increase when the greedy receiver also modifies RTS and data frames corresponding to TCP ACK packet.

5.2 Misbehavior 2: Spoofing ACKs

We evaluate misbehavior 2 using TCP traffic since this misbehavior targets TCP. We use a 4-node topology (2 senders sending to 2 receivers), where all the wireless nodes are within communication range of each other for evaluation. The loss rates on all wireless links (e.g., NS-NR, GS-GR, NS-GR and GS-NR) are the same.

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). Fig. 7 shows the goodput of both receivers when one of them misbehaves versus when neither misbehaves. The x-axis shows both bit error rate and the corresponding data frame error rate. 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. Moreover, 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 TCP goodput. In an extreme, when the loss rate is

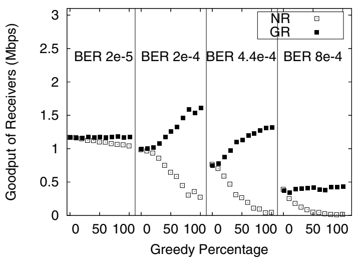


Figure 8. Goodput of two TCP flows NS-NR and GS-GR, when greedy percentage and loss rates vary.

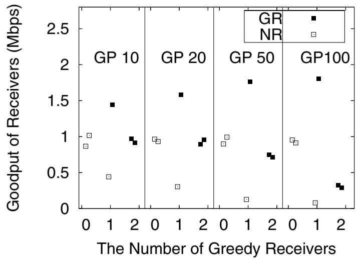


Figure 9. Goodput under 0, 1, or 2 greedy receivers (All flows use TCP, and BER=$2e^{-4}$).

high enough, both TCP flows get virtually zero goodput regardless of whether one misbehaves or not.

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). Fig. 8 summarizes the results. As we would expect, the goodput of greedy receiver increases with GP. 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 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 it also suffers degradation due to the 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 Fig. 9, 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. Fig. 10(a) compares the average goodput of a greedy receiver and normal receivers when they share one AP, and Fig. 10(b) shows the results when each receiver receive data from a different AP. As they show, in both cases the average throughput of greedy receiver is higher than that of the normal receivers. Moreover, the goodput difference between the greedy and normal receivers is larger under multiple APs due to lack of head-of-line blocking.

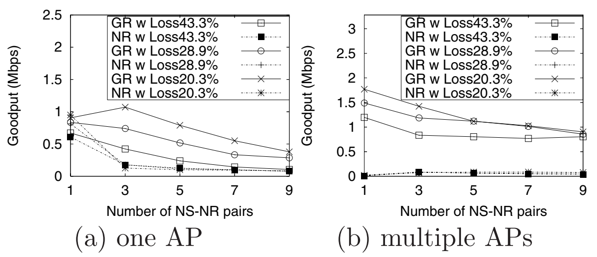


Figure 10. One greedy receiver competes with a varying number of NS-NR pairs.

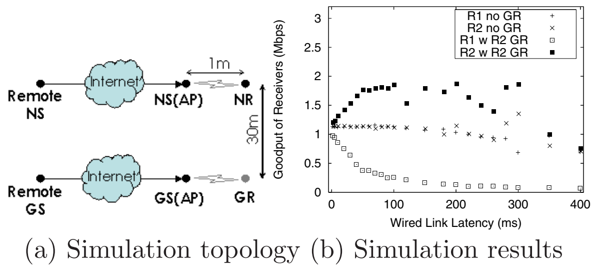


Figure 11. Goodput under remote TCP senders, where both wireless links to the greedy and normal receiver have BER=$2e^{-5}$.

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 Fig. 11(a). We vary the wired link latency from $2ms$ to $400ms$, and set BER of both wireless links to $2e^{-5}$. Fig. 11 compares goodput under no greedy receiver versus under one greedy receiver ($R2$ is a greedy receiver in this case). We observe that increasing wireline latency initially increases the gap between the normal 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 its goodput as delay increases, and the goodput gain from the normal receiver is not enough to offset such drop.

5.3 Misbehavior 3: Sending Fake ACKs

For misbehavior 3, a greedy receiver sends an ACK even upon receiving a corrupted data frame. This misbehavior is effective when the greedy receiver uses UDP, and the source and the destination address in the corrupted DATA frame are preserved. As shown in Table 1, this is quite common. 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.

Vary greedy percentage: As shown in Fig. 12, an increasing greedy percentage increases the discrepancy

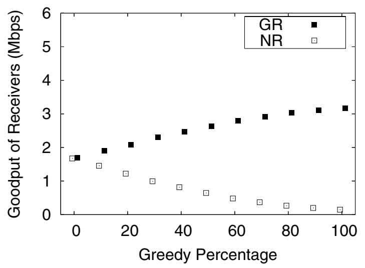


Figure 12. Goodput of two UDP flows NS-NR and GS-GR, when NS and GS are hidden terminals.

Data error rate	no GR (Mbps)		1 GR (Mbps)		2 GRs (Mbps)	
	$NR1$	$NR2$	NR	GR	$GR1$	$GR2$
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 2. Goodput of two receivers under 0, 1, or 2 greedy receivers.

between the goodput of the normal and greedy receivers. When GP=100% (i.e., greedy receiver fakes ACK on every corrupted data packet), the greedy receiver shuts off the other connection. This is because faking ACKs makes GS's contention window (CW) considerably smaller than NS's CW.

Vary the number of greedy receivers: Next we compare the performance when both receivers are greedy. Interestingly, the performance would depend on the types of losses. Under congestion-related losses due to hidden terminal, both greedy receivers suffer. Their goodput is reduced by 48%. This is because faking ACK effectively disables exponential backoff in 802.11 and senders send faster than they should, creating more collisions. In comparison, when the loss is non-congestion related (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 2. This is because under non-congestion loss, 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 non-congestion loss, faking ACKs can be considered as a useful surviving technique. However, this is not recommended under congestion losses.

Vary the number of sender-receiver pairs: Finally we consider one greedy receiver competes with a varying number of normal receivers, where all of them

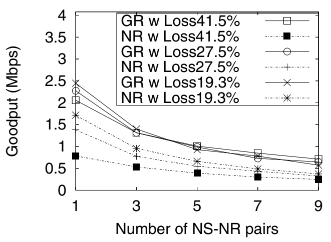


Figure 13. One greedy receiver competes with a varying number of NS-NR pairs.

experience the same loss rates. As shown in Fig. 13, 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.

6 Evaluation in Testbed

Next 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). They form two senders and two receivers. The locations of the nodes are fixed, on the same floor of an office building. Each node runs Fedora Core 4 Linux, and is equipped with 802.11 a/b/g NetGear WAG511 using MadWiFi. In our experiment, we enable RTS/CTS, and use a fixed 6 Mbps as MAC-layer data rate. 802.11a is used in our testbed experiments to avoid interference with campus 802.11b wireless LAN in the building.

Our testbed evaluation focuses on misbehavior 1, since MadWiFi currently does not allow us to implement the other two misbehaviors. Given the trend of moving more functionalities to software, this is not an inherent constraint. We implement misbehavior 1 as follows. Because the current MadWiFi does not allow us to directly modify NAV in CTS frames, we get around this problem by implementing RTS inflation on one of the senders. We increase RTS NAV to $32700\mu s$. This automatically triggers inflated NAV in CTS frames. (The inflated CTS NAV is $32655\mu s$.) Since we want to study the impact of a greedy receiver, we have the sender transmit at lowest power so that its RTS with inflated NAV is not overheard by the other sender and receiver. Only the CTS frames from the greedy receiver is heard by all the other nodes to effectively create greedy receiver misbehavior.

Table 3 compares the goodput under 0, 1, or 2 greedy receivers. The reported goodput is median over 5 runs, where each run lasts 2 minutes. As it shows, without greedy receiver, both receivers get similar goodput. When only one receiver is greedy, the greedy receiver gets virtually all the bandwidth and starves the normal receiver. When both receivers are greedy, the one transmitting earlier dominates the medium and starves the other receiver. These results

Transport	no GR (Mbps)		1 GR (Mbps)		2 GRs (Mbps)	
	$NR1$	$NR2$	NR	GR	$GR1$	$GR2$
UDP	3.06	2.43	0.04	4.65	0	4.64
TCP	2.31	2.36	0.01	4.43	0	4.37

Table 3. Testbed evaluation of NAV inflation.

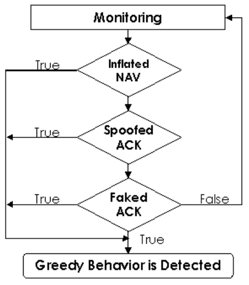


Figure 14. Detecting greedy receivers.

are consistent with simulation results, and confirm the serious damage of greedy misbehavior in real networks.

7 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. 14 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.

7.1 Detecting Inflated NAV

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

7.2 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

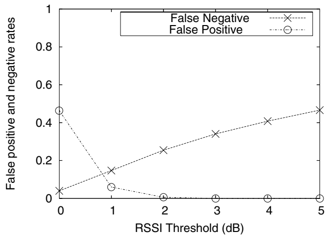


Figure 15. False positive and false negative vs. RSSI threshold.

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 RSSI measurements from our testbed, consisting of 16 nodes spread over one floor of an office building. Our measurements show that around 95% RSSI measurements differ from median RSSI of that link by no more than 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. 15 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.

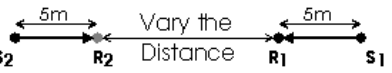
The previous detection is effective when RSSI from NR is relatively stable and RSSI from GR is different from NR. To handle highly mobile clients, which experience large variation in RSSI, the sender can use a cross-layer approach to detect the greedy behavior. For each TCP flow, it maintains a list 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.

7.3 Detecting Faked ACKs

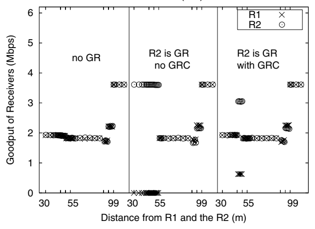
To detect greedy receivers that send MAC-layer ACKs even for corrupted frame, the sender compares the MAC-layer loss with the application layer loss rate. The latter can be estimated using active probing (e.g., ping). Since packets are corrupted, GR cannot send ping response and we can measure the true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

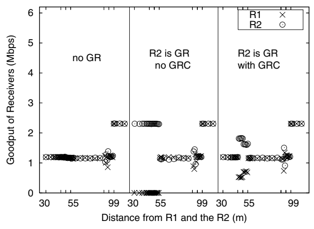
8 Evaluation of Detection

We implement in NS-2 the greedy receiver countermeasure (GRC) against inflated NAV and ACK spoofing described in Section 7.



(a) Evaluation topology





(b) UDP goodput (c) TCP goodput

Figure 16. GRC effectively detects and mitigates inflated CTS NAV.

First we evaluate the countermeasure against inflated CTS NAV using the the topology in Fig. 16(a), where communication and interference ranges are 55m and 99m, respectively. Fig. 16(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 Fig. 16(c).

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 Fig. 17, 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.

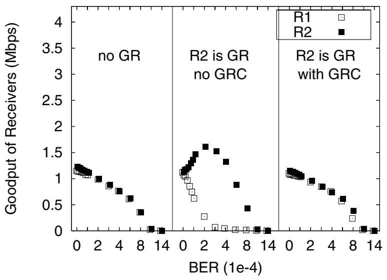


Figure 17. GRC effectively detects and recovers from ACK spoofing under varying BER.

9 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.

References

[1] 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.
[2] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On selfish behavior in CSMA/CA networks. In *Proc. of IEEE Infocom*, Mar. 2005.
[3] IEEE Computer Society LAN MAN Standards Committee. *IEEE 802.11: Wireless LAN Medium Access Control and Physical Layer Specifications*, Aug., 1999.
[4] 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.
[5] In-stat. `http://www.in-stat.com/catalog/Wcatalogue.asp?id=167`.
[6] Revenue from wireless hotspots. `http://blogs.zdnet.com/ITFacts/index.php?blogthis=1&p=9319`.
[7] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *Proc. of ACM MOBICOM*, Sept. 2002.
[8] 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.
[9] P. Kyasanur and N. Vaidya. Detection and handling of MAC layer misbehavior in wireless networks. In *IEEE Transactions on Mobile Computing*, Apr. 2004.
[10] The network simulator – ns-2. http://www.isi.edu/nsnam/ns/.
[11] 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.
[12] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson. TCP congestion control with a misbehaving receiver. *ACM Computer Communications Review*, Oct. 1999.
[13] D. Tang. Analysis of a local-area wireless network. In *Proc. of MOBICOM*, Sept. 2000.
[14] 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.