Stealing Bandwidth via Link-targeted Interference Attacks*

MINA GUIRGUIS msg@cs.bu.edu
AZER BESTAVROS best@cs.bu.edu
IBRAHIM MATTA matta@cs.bu.edu

Computer Science Department
Boston University
Boston, MA 02215

JULY 8, 2004

Abstract

We expose an adversarial attack scheme that aims to steal bandwidth for the benefit of a particular set of flows through lunching a distributed interference attack streams on competing flows. The extent to which the interference attack streams were successful in reducing or denying bandwidth from competing flows determines the amount of bandwidth stolen. Given such a goal, our exposed scheme stands in sharp contrast to sustained high-rate Denial-of-Service (DoS) attacks targeted directly at a specific resource or a set of flows. We demonstrate two schemes for the construction of an interference attack stream that would evade detection, and thus challenging counter-DoS techniques. Our results show the vulnerability of the current Internet to those new forms of attacks that could be easily mounted with a few number of zombie clients. We validate our findings through simple analysis, simulations and real Internet experiments.

Keywords: Internet; Denial-of-Service (DoS) Attacks; TCP; Performance Evaluation.

1. Introduction

This paper exposes a distributed interference attack scheme that would compromise a set of links so as to provide additional bandwidth to a particular set of flows. Given such a goal, this scheme stands in sharp contrast to lunching attacks just for the sake of shutting off some flows [12] or crashing a particular web site [3].

DoS [4] and DDoS [7] attacks have become a major threat for almost every Internet service. Recently, MyDoom crashed SCO Group's web site through lunching a DDoS attack that involved 100K-200K zombies; MyDoom had cost the global economy over $26.1B [15]. Having compromised that huge number of zombie clients, the possibilities for the kind of damage are abound.

Our exposed scheme for illegitimately giving a particular set of flows (we refer to them as *supported flows*) additional bandwidth relies on limiting the throughput of other flows that would potentially compete with the supported flows over a limited amount of bandwidth. Such limitation is achieved through attacking other links (we refer to them as *victim or target links*). Clearly, victim links are not being traversed by the supported flows.

When the supported set of flows share an Internet link with other flows, they get their fair-share of resources as would be allocated by the widely used Transmission Control Protocol (TCP) [6]. As more bandwidth becomes available, TCP's probing mechanism through its additive-increase would enable flows to utilize such bandwidth, keeping the network operating at high utilization. At the same time, experiencing packet losses would trigger TCP's multiplicative-decrease where the window is halved to alleviate congestion.

The interference attacks that we envision would cause competing flows to experience different levels of packet losses. Hence, their throughput would decrease, limited by the damage as opposed to their network fair-share, yielding a slack of resources. This slack of resources would be naturally acquired by the supported flows, causing them to acquire more bandwidth allocation. As will be discussed throughout the paper, depending on the feasibility of the attack in terms of identifying potential links to attack and the magnitude of the attack, different bandwidth allocations could be (illegitimately) provided to the set of supported flows.

Figure 1 depicts our envisioned setup; we assume the presence of a number of zombie clients scattered around the globe. These clients are controlled by an attack controller.[1]

An entity responsible for a set of flows illegitimately requests the aid of an attacker, possibly through out-of-band communication, providing the attack controller with the Internet path the set of supported flows are traversing. The

*This work was supported in part by NSF grants ANI-0095988, ANI-9986397, EIA-0202067 and ITR ANI-0205294, and by grants from Sprint Labs and Motorola Labs.

[1] These machines are usually compromised by a virus or a worm. It is outside the scope of this paper how this setup was formed. Suffices to say that recent attacks in [7] were all carried out by similar setups.

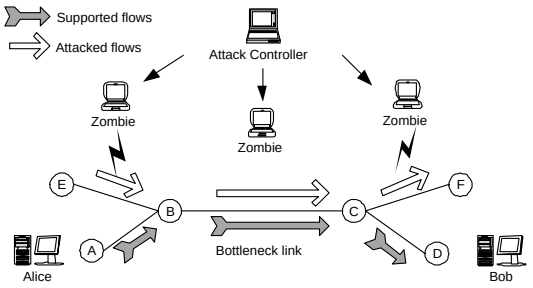

Figure 1: Adversarial scheme we envision

attack controller, in return, would broadcast such information to the zombie clients. The zombie clients would run a Network Inference Algorithm that would enable them to identify a set of target links and a set of destinations through which their interference traffic is sent. Figure 1 depicts one of possibly many scenarios where the supported flows are traversing between Alice and Bob through the path $A - B - C - D$. The bottleneck link in this case is BC and the throughput between Alice and Bob is limited by the fair-share given at the bottleneck link. The zombie clients would then identify links EB and CF as potential target links. Interference with the traffic on links EB and CF, would limit the achievable throughput for flows on those links, causing Alice and Bob to receive more than their fair share of resources on link BC. It is worth mentioning that it could be the case that no victim links could be identified by the zombies, hence lunching the attack is not carried out. We will present simple analysis showing that this will not happen with a high probability, providing a lower bound on the required number of zombie clients.

Evading detection is one of the most important challenges faced by the attack controller as well as by the zombie clients; we identify two possible schemes for evading detection. Such schemes rely on timing orchestration while consuming a fairly low intensity traffic per attack stream. Hence, the attack traffic may not be detected by regular counter-DoS techniques. More to the point, the challenge with this new form of interference attack is that it does not require a complete denial of service on the victim links, but rather it only aims to limit the aggregate bandwidth of victim flows.

Paper Outline: The rest of the paper is organized as follows. Section 2 describes our exposed scheme in detail. Performance evaluation is presented through simulations and real Internet experiments in Section 3. We revisit related work in Section 4. Section 5 concludes the paper.

2. Attack Orchestration

2.1 Interference Attack Construction

End-system protocols (*e.g.,* TCP) rely on feedback mechanisms to adapt their sending rates to match their "fair share" of network resources. TCP reduces its sending rate on packet loss/marking and increases its rate on successful packet transmission. Typically, the decrease in rate, which is needed to protect against congestion collapse, is drastic—*e.g.,* by halving the sending rate—whereas the increase in rate, which is needed to probe for available bandwidth, is slow—*e.g.,* by linearly increasing the sending rate over time. Additive-Increase-Multiplicative-Decrease (AIMD) rules[2] ensure that flows react adequately to congestion in a "friendly" manner to one another—hence the TCP-friendly label [9]. Moreover, these protocols react even more swiftly to excessive losses by completely shutting off their sending rates for a long period of time (*e.g.*, timing out in TCP).

The adaptation strategies of transmission control protocols such as TCP, while crucial for alleviating congestion, make them vulnerable to losses that are generated through other processes—namely losses that are not the result of persistent congestion (*e.g.,* wireless losses). The impact of such losses on TCP performance was considered in many studies; examples include [1]. In these studies, however, the processes interfering with TCP's adaptation could be considered "non adversarial" in the sense that the losses were more or less the result of (say) a random process as opposed to a calculated attack. Indeed, in recent work, it was shown that an attacker could potentially shut off the communication between two parties (*e.g.,* Alice and Bob) by mounting what is termed as a "shrew" attack [12]—an attack that exploits TCP's time-out mechanism, which is how TCP adapts to persistent congestion.

As we hinted in the introduction and as the results in this paper will demonstrate, illegitinately giving a particular set of flows additional bandwidth doesn't require a complete shut-off of the competing flows, but rather only *limiting* their achievable throughput. Hence, lunching a "shrew" attack would be an over-kill, not to mention the suspicious behavior it may cause. We consider an interference attack comprising a burst of M packets (or bytes) transmitted at the rate of δ packets (or bytes) per second over a short period of time τ, where $M = \delta\tau$. This process is repeated every T units of time. We call M the *magnitude* of the attack, δ the *amplitude* of the attack, τ the *duration* of the attack, and T the *period* of the attack. Typically τ should be smaller than

[2]Other TCP-friendly increase/decrease rules have also been proposed and evaluated [2]. All would be susceptible with various degrees to the same issues we consider in this paper.

T. Thus the interference traffic, $I(t)$, at time t is given by:

$$I(t) \quad = \quad \begin{cases} \delta & t \bmod T \leq \tau \\ 0 & otherwise \end{cases} \qquad (1)$$

Such a square-wave attack traffic is advantageous to an attacker in two ways: First, it provides the freedom of varying the attack parameters (δ, τ and T) causing different levels of damage; Second, it allows the zombies' traffic to go unnoticed through having an average attack traffic $\frac{M}{T}$ that is much lower than the peak rate δ. This gives one degree of freedom to evade detection. A second degree of freedom is addressed in the next subsection, where the traffic $\frac{M}{T}$ would be distributed across different attack streams.

It is worth mentioning that it is the same AIMD mechanism that the interference attacks exploit, *is* the one that allows the supported flows to take advantage of available bandwidth once victim flows back-off.

2.2 Selecting the Targeted links

In this subsection, we turn our attention to how the interference attacks can be routed through the network. In particular, how can the zombie clients identify the potential target links through which they should send their traffic. Potential target links are those links that are more likely to carry traffic that is competing with the supported flows. Figure 2 depicts a more detailed view of Figure 1, where Alice and Bob are communicating through the path $A - B - C - D$ and are limited by their fair-share at link BC.

Figure 2 shows a subset of four potential target links (GB, EB, CF and CH) that could carry traffic through link BC. Notice that the problem of identifying those potential target links is the same as discovering the neighborhood (and interfaces) of both routers of the bottleneck link. We present a three-step algorithm that would allow the zombie clients to discover the area around each router with high probability. Then, we give a lower bound on the number of zombie clients that would cause a complete discovery of the neighborhood.

Initially, each zombie client receives the common path traversed by the supported flows from the attack controller. For ease of presentation, we assume they only receive the IP addresses of both routers along the path of the supported flows that represent the bottleneck link. In the first step of our algorithm, each zombie client would trace the route to those two routers. This will lead to the discovery of those routers along the path between the zombies and the two initial routers. This step corresponds to the discovery of routers E, F, G and H in Figure 2. In particular, zombie $Z1$ would know about E and F, while zombie $Z2$ would know about G and H. We refer to this list of routers as the "previous-hop" list.

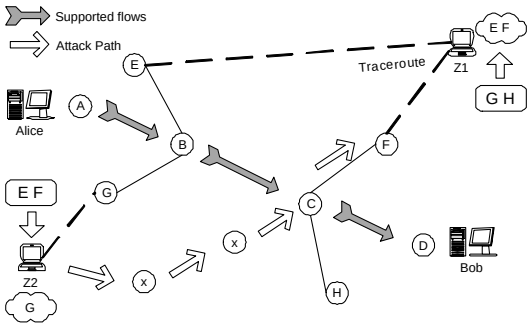

Figure 2: A more detailed view of the adversarial scheme we consider

In the second step, the zombie clients exchange their "previous-hop" lists. Through exchanging these lists, each side of the zombies would know about the routers located on the other side of the bottleneck link. Exchanging these lists are made possible through the attack controller. Decentralized approaches are also feasible but would require more work from the zombie clients. This step corresponds the discovery of routers E and F by zombie $Z2$ and routers G and H by zombie $Z1$ as illustrated in Figure 2.

The third and final step is another traceroute, now to the opposite side (e.g. zombie $Z2$ traceroutes to routers E and F), to remove any paths that contain any segment over which the supported flows traverse. Thus avoiding a scenario where the attack traffic is traversing along the same paths as the supported flows. The destination to be used for the attack traffic could by any valid IP address that belongs to a target router. For example, zombie $Z2$ sends its attack traffic to router F as illustrated in Figure 2—Given that some flow(s) competing with the supported flows and traversing CF back-off due to losses from the attack, the supported flows can then acquire their bandwidth. In practice, the longer the path of the supported flows, the more chance the zombie clients have to identify more potential target links.

Notice that an obvious attack is to send interference traffic destined directly to the bottleneck routers. However, this would result in overloading those routers as they process these attack packets rather than simply forwarding them, thus negatively impacting the performance of the supported flows. The above inference algorithm targets links that are not on the path of the supported flows.

2.3 A Lower Bound on Zombies

Having described what constitutes an interference attack and how they are being routed through the network, we show that such attacks are feasible with a small number of

zombies. Let the number of zombies be denoted by N and assume that there is only one router of degree d that we would like to discover its neighborhood (i.e. the d adjacent routers). We are interested in finding the minimum number of zombies that would cause the discovery of the d neighbors with high probability. In particular, the probability of missing an interface after N traceroutes from the N zombies,[3] assuming uniform distribution among the router's d neighbors, is given by:

$$(1 - \frac{1}{d})^N \approx e^{-\frac{N}{d}} \quad (2)$$

Fixing the above probability to α, a lower bound on N, N^- is given by:

$$N^- \quad = \quad -d \times ln(\alpha) \quad (3)$$

For any distribution of traffic, equation (2) can be modified to:

$$(1 - P_{min}^d)^N \approx e^{-\frac{N}{P_{min}^d}} \quad (4)$$

where P_{min}^d is the minimum probability over all d interfaces.

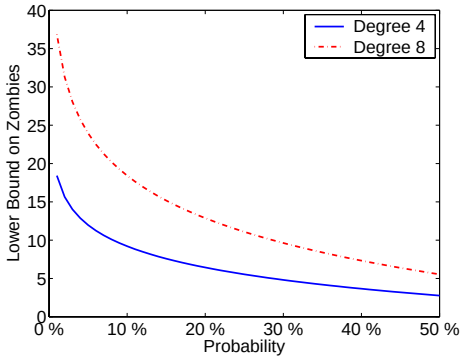

Figure 3: Lower bound N^- for different probabilities and degrees 4 and 8 using uniform distribution

Figure 3 shows the lower bound N^- as a function of changing α, and degrees $d = 4$ and $d = 8$ assuming a uniform distribution among a router's interfaces. One can see that even for a small probability of 1%, the minimum number of zombies is very small (less than 20 for the case of $d = 4$ and twice as much for $d = 8$).[4]

Going back to our second degree of freedom for evading detection— that of spoofing the sources and *destinations* IP addresses. Unlike traditional DoS, every attack traffic packet could be sent from a source to a different destination. Moreover, as long as they are known to be routed through the resource under attack, these destinations do not even have to be legitimate or live addresses. As far as a detection mechanism in the middle could tell, the packets going through are between different sources and destinations. All these packets could be produced by a single zombie client. However, dividing the attack magnitude over few zombies would be less suspicious at the ingress link the zombie is connected to.

[3]Notice that this problem is the same as throwing M balls in K bins problem. The same analysis could be applied.

[4]Notice that in Figure 3, the line corresponding to a degree d would represent the same lower bound on the number of zombies for any distribution of traffic over interfaces if the minimum probability across the interfaces is $P_{min}^d = \frac{1}{d}$.

3. Performance Evaluation

In this section, we present results from ns-2 [8] simulation experiments we conducted to assess the effect of DoS and interference attacks on other flows, for improving the bandwidth allocated to the supported flows. We then validate our simulation results through live Internet Experiments performed inside our laboratory.

3.1 Simulations

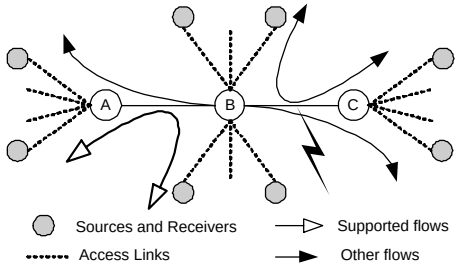

Figure 4: The two-link topology used in ns-2 simulation experiments.

Figure 4 depicts the topology under consideration. We have two links AB and BC of capacity 100 Mbps each. All links have a one-way propagation delay of 1 msec. A total of 20 FTP connections, with unlimited data to send, traverse the topology from A to C. We refer to these as the AC flows. In addition, two groups of 10 FTP connections each traverse exactly one of the links in the topology. We refer to these as the AB and BC flows. Sources as well as receivers of these FTP flows connect to the routers A, B and C through access links. RED is used as the queue management at the links AB and BC. We set RED's minimum and maximum buffer thresholds to 50 and 120 packets, respectively. The weight parameter β was set to 0.0001 and P_{max} was set to 0.1. The buffer size is chosen to be 250 packets at each link. All packets are 1,000 bytes in size. Since flows AC traverse two bottleneck links, they tend to have less throughput than flows AB and BC. So we adjusted the

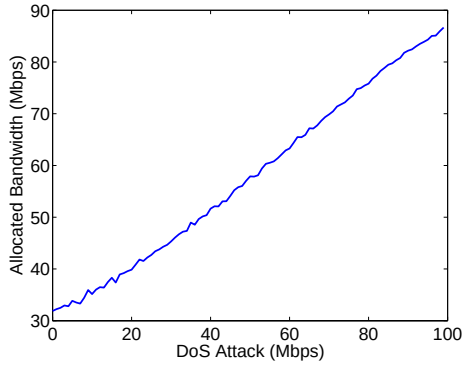

Figure 5: Improvement in allocated bandwidth as the level of DoS attack increases

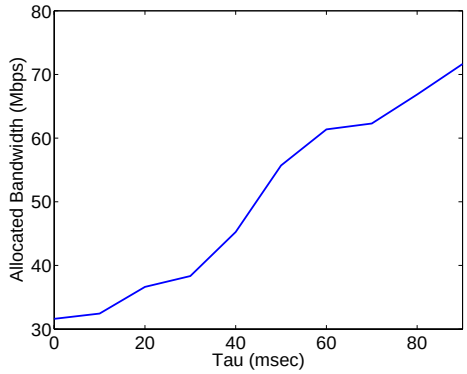

Figure 6: Improvement in allocated bandwidth as τ changes for a fixed T of 0.5 and δ of 110 Mbps

propagation delay on the access links so that all connections have the same round-trip time. The attack traffic traverses through link BC. Our goal is to interfere with the flows AC for an improved allocated bandwidth for flows AB.

Figure 5 represents our first experiment, where the attacking sources use regular sustained high-level of DoS streams. One can see that the improvement in throughput allocated to flows AB increases linearly with the magnitude of the attack. Clearly, this is a feasible way to attack, but it has the drawback of sending a lot of attack traffic. It is worth mentioning that this form of attack still could evade detection using spoofed sources and destinations, but still it is not considered a low-rate attack.

Our next experiment improves the previous result significantly where we put the interference attacks we expose into play. We adjusted the attack period T to 0.5 seconds, the attack rate δ to 110 Mpbs and we varied the attack duration τ from 0 (no interference attack is lunched) to 90 msec. Figure 6 shows the effect of τ on the allocated throughput for flows AB. As τ increases, the bandwidth allocated to flows AB increases. Notice that doubling the bandwidth for flows AB from 31 Mbps to 60 Mbps required a DoS attack of an average rate of 60 Mbps. However, for a τ value of 60 msec, the same bandwidth allocation was achieved for an average attack traffic of 11 Mbps. Almost a factor of 6 improvement!

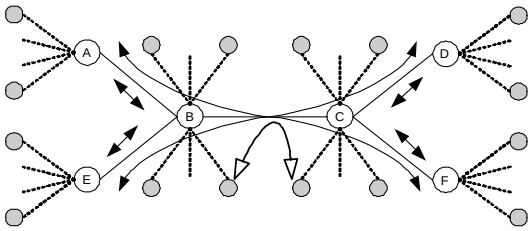

Figure 7: The five-link topology used in ns-2 simulation experiments.

We next turn our attention to a topological setting that would likely arise through our exposed DoS scheme, where the zombie clients will be injecting interference streams on the ingress points of some router and the egress points of another router. Figure 7 depicts such a topology. We have five links AB, EB, BC, CD and CF of capacity 100 Mbps each. All links have a one-way propagation delay of 1 msec. A total of 20 FTP connections, with unlimited data to send, traverse the topology from A to D and from E to F. In addition, five groups of 10 FTP connections each traverse exactly one of the links in the topology. We refer to these as the AB, EB, BC, CD and CF flows. RED is used as the queue management on all the links and is parameterized as above. Sources as well as receivers of these FTP flows connect to the routers E, A, B, C, D and F through access links of propagation delay of 8 msec. We consider an attack whose goal is to improve the bandwidth allocated to BC flows, through interfering with flows AD and EF. The zombie clients, after running their inference algorithm, will discover the links AB, EB, CD and CF.

Figure 8 shows the different bandwidth allocations to flows BC, from ID 1 to 10, as the inteference attack was succesful in identifying and hiting one link (AB), two links (AB and EB) and the four links (AB, EB, CD and CF). All attack traffic followed the same paramters of τ = 50 msec, δ = 110 and T = 0.5 sec. Two things to note here; First, when the interference attack was lunched on link AB, the slack bandwidth was divided between the BC and the EF flows, thus the improvement is not very high (from a total of 15 Mbps under no attack to 22 Mbps). Once the link EB is attacked, the improvement was much more pronounced (up to 43 Mbps). Second, the improvement from attacking the four links didn't buy as much improvement as attacking two links (now BC flows can get up to 46 Mbps). This is due to hitting the same flows on two links. The total average attack traffic on any attacked link was around 9.5 Mbps.

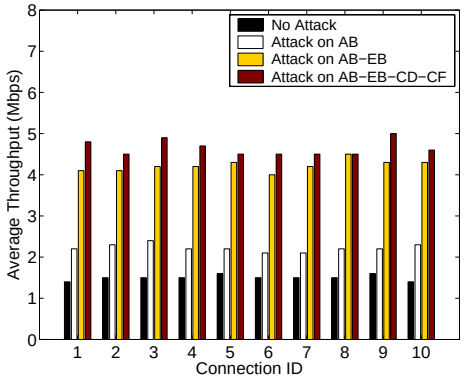

Figure 8: Throughput allocated to each flow from the BC flows.

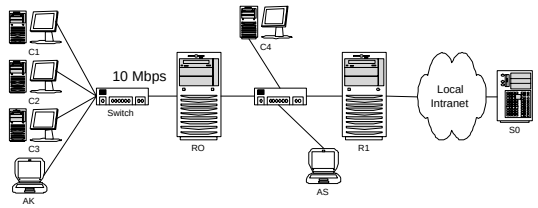

Figure 9: The experimental setup used to conduct our Internet experiments

This means having three zombies is enough to ensure that the attack traffic from each consumes less than its resource fair-share, thus evading detection. This is because each 100 Mbps link, except for BC which is shared by 50 flows, is 100 Mbps shared by 30 flows, thus a fair share of 3.3 Mbps per flow.

3.2 Internet Experiments

Figure 9 depicts the experimental setup we used for our Internet experiments. It consists of two routers ($R0$ and $R1$), a local content server ($S0$), four client machines (C_1, C_2, C_3 and C_4), a source of attack traffic (A_s) and a sink of attack traffic (A_k). The router's server-side interface is connected to a 100 Mbps switch that connects to the content server machine ($S0$) and the attack source (A_s). The router's client-side interface is connected to another 100 Mbps switch that connects it to the local subnet where client machines (C_1, C_2, and C_3) and attack sink (A_k) reside. The network interface cards on all machines run at 100 Mbps except for the router's client-side interface, representing the bottleneck link, which runs at 10 Mbps. All machines run Linux RedHat version 2.4.20. The router uses iproute2 and tc [11] to run different packet scheduling disciplines. In all experiments we report on in this paper, we used a packetized version of FIFO (called pfifo). A client (C_i) is configured to

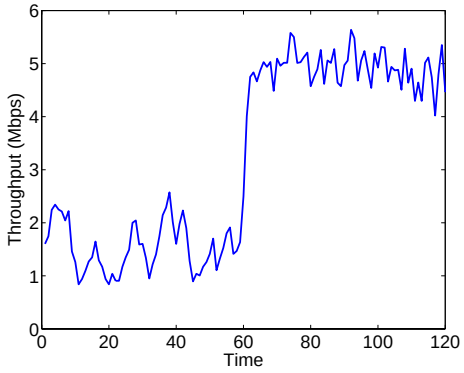

Figure 10: Throughput allocated to connection $C4$-$S0$. Attack is lunched at time 60 between A_s and A_k, parameterized by δ = 9.5 Mbps, τ = 60 msec and T = 0.5 sec. Average attack traffic passing through $R0$ is 0.6 Mbps

request local data transfers from the local server $S0$. As described in Section 2, the attack source (A_s) injects UDP packets destined to sink (A_k)[5] following the square wave pattern with parameters δ, τ and T.

In this experiment, each of the 4 clients opens a TCP connection to the server $S0$ for a total of 3 TCP flows traversing through $R0$ and 4 TCP flows traversing through $R1$. $R1$ was the bottlneck for all the flows. Consider that the TCP flow from C_4 to $S0$ requests additional bandwidth from the attacker. This in return triggers an interference attack through $R0$ to cripple the other 3 TCP flows. Figure 10 shows how the throughput allocated between C_4 and $S0$ improves once the attack is lunched at time 60. This attack had a value of τ of 60 msec, T of 0.5 sec and δ of 9.5 Mbps. The average attack traffic was 0.6 Mbps.

4. Related Work

DoS attacks [4, 3] and its many variants [5] could be characterized as targeting one dimension of a system's service quality–namely, its availability. There are a number of papers that classify various forms of DoS attacks; examples include [10, 14, 13]. In this paper, we have focused on attacks whose perpetrators are not focused on denying access, but rather interfering with other competing flows. More importantly, in this paper, we have focused on the harder-to-detect, low-intensity attacks, *i.e.*, with modest aggressiveness compared to the aggressiveness required for DoS attacks. The "Shrew" attack proposed in [12] is an example of a low-intensity, harder to detect attack which is targeted at

[5]Unlike traditional DoS attacks, A_k is not the target of the attack, but rather a bystander which does not even have to be on-line (as long as packets destined to it are routed through the target of the attack—namely router $R0$).

a subset of flows going through a network link, with the intension of shutting off these flows by synchronizing the attack traffic in such a way to cause these flows to perpetually timeout. For the purpose of this paper, the "shrew" attack would be an over-kill, not to mention the suspicious concern it may trigger. Moreover, it can't be tuned to achieve different levels of damage. Our interference attacks could be tuned, through adjusting the parameters, to cause the minimum damage possible for achieving its goal, that of providing additional bandwidth to a set of flows by stealing it from competing flows.

5. Conclusion

In this paper, we have exposed an adversarial scheme capable of providing additional bandwidth to a particular set of flows by stealing it from competing flows. This is done by sending enough attack traffic to interfere with competing flows on links unused by supported flows. Our results show that such attacks could be orchestrated with very few number of zombie clients while evading detection. We believe that shedding light on such vlunerabilities and how they can be exploited is crucial as it motivates the need for the development of more resilient mechanisms towards these new forms of attacks.

Acknowledgment: We would like to thank Jeffrey Considine for his comments and fruitful discussions on this work.

References

[1] Hari Balakrishnan, V. Padmanabhan, S. Seshan, and R. Kartz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. In *Proceedings of ACM SIGCOMM'96*, 1996.

[2] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *Proceedings of INFOCOM'2001*, 2001.

[3] CERT Coordination Center. CERT Advisory CA-1996-21 TCP SYN Flooding and IP Spoofing Attacks Original issue date: September 19, 1996. http://www.cert.org/advisories/CA-1996-21.html.

[4] CERT Coordination Center. Denial of Service Attacks. http://www.cert.org/tech_tips/denial_of_service.html.

[5] CERT Coordination Center. Trends in Denial of Service Attack Technology, October 2001. http://www.cert.org/archive/pdf/DoS_trends.pdf.

[6] D.M. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Computer Networks and ISDN Systems*, 17:1–14, 1989.

[7] *Distributed Denial of Service (DDoS) Attacks/tools* . http://staff.washington.edu/dittrich/misc/ddos/.

[8] E. Amir et al. UCB/LBNL/VINT Network Simulator - ns (version 2). Available at http://www.isi.edu/nsnam/ns/.

[9] The PSC Networking group. The TCP-Friendly Website. http://www.psc.edu/networking/tcp_friendly.html.

[10] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 99–110. ACM Press, 2003.

[11] *iproute2 and tc*. http://snafu.freedom.org/linux2.2/iproute-notes.html.

[12] A. Kuzmanovic and E. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks (The Shrew vs. the Mice and Elephants). In *Proceedings ACM SIGCOMM'03*, karlsruhe , Germany, August 2003.

[13] C. Meadows. A formal framework and evaluation method for network denial of servic. In *Proceedings of the 12th IEEE Computer Security Foundations Workshop*, June 1999.

[14] J. Mirkovic, J. Martin, and P. Reiher. A Taxonomy of DDoS Attacks and DDoS Defense Mechanisms. *Technical report 020018 Computer Science Department, University of California, Los Angeles* .

[15] The Salt Lake Tribune. As forecast, worm takes SCO offline (February 2, 2004). Available from http://www.sltrib.com/2004/Feb/02022004/utah/134908.asp.