Using Packet Symmetry to Curtail Malicious Traffic

Christian Kreibich, Andrew Warfield,
Jon Crowcroft, Steven Hand, Ian Pratt
University of Cambridge Computer Laboratory
{firstname.lastname}@cl.cam.ac.uk

1 Introduction and Motivation

What is "bad" traffic? While even non-technical home users have become familiar with terms such as "denial of service" and "worm outbreaks", a technical solution to these forms of malicious traffic continues to elude researchers and network administrators alike. The reason for these difficulties is also rather well understood: The network itself simply has no such notion of bad traffic.

This paper argues that a key omission from the original Internet architecture was that of *packet dynamics*. The historical obsession with end-to-end design has resulted in connection signalling, flow- and congestion-control loops being pushed to the transport layer (if not simply ignored); these decisions have effectively legislated *for* the sort of resource exploitation attacks that are a problem today. We argue that while end-to-end design is vital to maximise freedom to innovate, the network must enforce a higher degree of mutual consent between communicating hosts.

As one example, today's server farms provide enormous CPU and bandwidth resources to essentially anyone, often leaving the site operators with the first line of responsibility in case of abuse. In this paper we investigate a *simple*, incrementally deployable modification of the existing architecture that benefits such sites *immediately*, while providing increasing protection to all members of the Internet as deployment becomes more pervasive. While many recent proposals have outlined major architectural changes [1, 2, 3], often including the introduction of explicit signalling, we have been interested in a network-level solution similar to TTL that leaves higher-level semantics largely unchanged. This thought exercise has lead us to the following two rather uncontentious observations:

1. **Implicit signalling.** It is much easier for a receiver to indicate the traffic that they do like, than the traffic that they do not. This signalling is done implicitly, simply by replying to received messages.
2. **Ingress limiting.** The most desirable location to detect and act against malicious traffic is at, or very near the source. This avoids resource consumption on intermediate links, as well as complexities introduced by source address spoofing [4, 5, 6].

We argue that *packet symmetry*, the ratio of transmitted to received packets, should become a fundamental principle of Internet protocol design: a high degree of packet symmetry embeds the notion of mutual consent within a protocol, allowing the receiver to implicitly throttle a sender by not replying to her packets. We further propose that symmetry be enforced on network transmissions at the edge: A simple enforcement mechanism may be placed in NICs or access providers' line cards, to delay or drop packets that result in strongly asymmetric communications. This extreme edge placement makes implementation easy, ensures a clear notion of packet provenance, and cannot be compromised by application or OS exploits on the end-host.

The remainder of this paper attempts to make the case for symmetry enforcement in the Internet. We do not yet claim to understand the exact parameterization of symmetry to enforce, in terms of the granularity of traffic, time, or the ratio of packets. However, we demonstrate a prototype implementation to illustrate the potential effectiveness of symmetry enforcement, and outline a detailed trace analysis of 24 hours of traffic to show the differentiation between symmetric and asymmetric traffic in the network today.

2 A Simple Idea

As discussed above, our approach is to interpret unsolicited traffic—traffic lacking in implicit acknowledgement from the receiving host—as being malicious. To classify packets we introduce the metric of *packet asymmetry*: we measure the number of transmitted packets (tx) and received packets (rx) per unit time and calculate the quantity:

$$\mathcal{S} = \log_e \left(\frac{tx + 1}{rx + 1} \right)$$

This metric produces negative values when rx outweighs tx, positive values when tx outweighs rx, and zero in the case of perfectly balanced traffic. The absolute value of $\mathcal{S}$ measures the magnitude of the asymmetry.

This metric has been carefully chosen for analysis: it allows an unbiased means of evaluating traffic, centred around zero, and compresses wildly asymmetric traffic ratios into a tractable range. While we were initially concerned that this metric might not be sufficiently sensitive, both measurement and initial implementation have shown it to be very useful to work with.

Given some network vantage point, the value of $\mathcal{S}$ may be calculated over traffic at some granularity (e.g. per-host, per-host-pair, per-flow) over a window of time. We may then take action against traffic that exceeds some threshold value of $\mathcal{S}$. The remainder of this paper is concerned with the effectiveness of this approach, and the selection of reasonable parameters for measurement and limiting. First though, it is worth detailing several design decisions inherent to our approach.

Measure packets, not bytes. Rather than comparing bytes transmitted in each direction, we simply count packets. With no knowledge of the internals of the data being sent, packets are much more likely to indicate the message structure that exists within a given protocol. Moreover, the implicit signalling to receive more data may be as simple as a TCP *ACK*, for which byte counts are considerably less useful than packets.

Measure and limit close to the transmitter. The outcome of the approach we advocate is that the policy of implicit signalling is enforced end-to-end: Receivers are responsible for generating sufficient backpressure on a channel to allow the transmitter to continue sending. Monitoring and enforcement, however, are performed within the network just outside the reach of the transmitting software (e.g. on a smart NIC). There are many reasons for this placement: First, we may clearly establish packet provenance, eliminating the need for trace-back [4, 5, 6]. Second, we eliminate all potential damage done to interior links as well as the target endpoint. Third, we minimize the aggregate amount of state that must be tracked, allowing a simpler implementation. And finally, by mandating that placement be near, but not within the transmitter's software stack, we are robust against exploits which circumvent the OS.

Delay, then drop. Unlike traditional IP congestion control, we opt to delay, rather than to drop packets. As asymmetry increases beyond a selected threshold, we introduce an increasing delay to the transmission of a queued packet. The intention is to be friendly to protocol congestion control approaches by more gently throttling transmitted packets. Where our approach is implemented in a smart NIC, queueing may be completely deferred to the OS. In a non-local device, for instance in an ISP, we anticipate queueing some number of packets for delayed transmission, and then beginning to drop.

The remainder of this paper attempts to show that this approach, although simple in nature, can be made to work. In the next section we detail a simple prototype acting on per-flow measurements. We then analyse a large trace of Internet traffic to establish how asymmetry might apply in general. The intention of this paper is to outline an overall approach that we hope to deploy in a college network within the next year; as such, feedback from the networking and systems communities would be very beneficial.

3 A Simple Prototype

As an initial proof-of-concept, we have implemented a naive traffic shaper based on the symmetry metric $\mathcal{S}$. Our implementation uses the `libipq` extensions for netfilter/iptables[1], the packet filtering framework for Linux.

Our prototype interposes on all traffic to and from the local host, calculating per-host-pair and per-flow asymmetry values. Based on our analysis, presented in Section 4, we have chosen to calibrate the filter to an asymmetry threshold of $\mathcal{S} = 2.0$. This corresponds to an approximately 8:1 ratio of transmitted to received packets within the window—a very liberal initial value.

Our prototype transparently forwards packets to the bottom half of the network stack providing that the current measured asymmetry is ≤ 2.0. When $\mathcal{S}$ exceeds this threshold, we begin to count packets as outstanding using a monotonically increasing packet counter n. Each outstanding packet is then delayed by $2^n ms$ before it is transmitted. This delay continues to accrue and be applied to transmit packets until the symmetry falls back below our threshold value, at which point the timer is reset. Note that both the asymmetry threshold and penalty are very weak in these examples. We anticipate that they could be set much more aggressively in a practical deployment.

Figure 1 shows an example of one class of traffic that we hope to eliminate. The graphs plot the transmit packet rate and the resultant asymmetry of a simple UDP flood attack against a remote host. The two hosts are connected over 100Mbit links in a local LAN. The local host transmits one KB UDP packets as fast as it can, saturating the link. The symmetry value plotted in the lower graph raises logarithmically as the attack continues.

Figure 2 shows the effect on the UDP flood. As can be seen, the UDP transmission is aggressively limited at the threshold due to the lack of received packets to offset the asymmetry. Compare this to the `scp`-based file transfer shown in Figure 3: `scp` also saturates the link (achieving a lower packet rate because it is sending larger packets), but stays well below the asymmetry threshold.

Despite the simplicity of our implementation, we feel that the results are very promising. Note that at a transmitting-host granularity, port scanning can be recognized in a manner almost identical to that used to limit the UDP flood. Instead, we would track transmissions to individual foreign host or (host, port) pairs.

[1] http://www.netfilter.org/

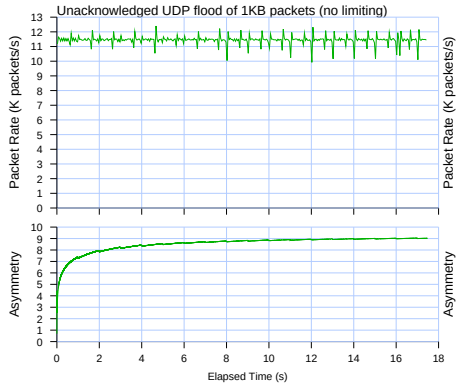

Figure 1: Simple DDoS Example: UDP Flood

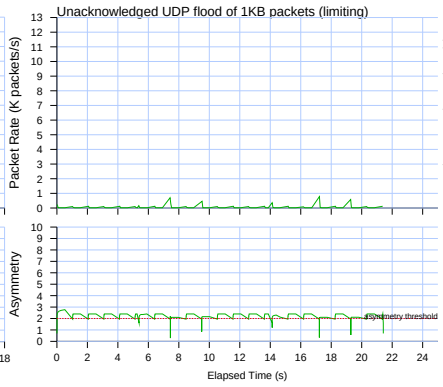

Figure 2: Symmetry-Limiting UDP Flood

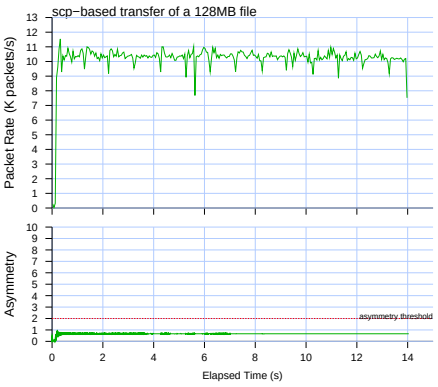

Figure 3: High Throughput TCP Unaffected

Packet asymmetry is clearly an effective approach for this simple pair of traffic examples, but is it useful in the general case? At what granularity should traffic be monitored? We now consider a trace-based analysis of Internet traffic to consider the applicability of our approach "in the wild."

4 Traffic Analysis

In order to determine what sort of traffic might be considered 'well-behaved', we performed analysis on a 24 hour packet trace collected at a non-university research institution. The trace captured every packet on the full-duplex Gigabit Ethernet link which connected the institution to the Internet. The trace contains over 573 million packets to/from over 170,000 IP addresses and totalling over 250 gigabytes of data—see [7, 8] for more details on the trace characteristics and the monitoring infrastructure used.

We have examined the degree of symmetry present in the trace data at several granularities: all traffic from each source host; traffic between host pairs; and finally per-flow traffic. The aim of this analysis has been to determine to what degree our symmetry metric can be used to distinguish well-behaved traffic, and how much state it might be useful to maintain in order to achieve this.

4.1 Host Packet Symmetry

We first examined symmetry from the point of view of each of the 170,000 individual hosts in the trace. We calculate $\mathcal{S}$ for all packets relating to that host at a variety of time scales, from one second up to one day. The intention of this measurement is *(i)* to characterize the ranges of symmetries that are exhibited within the trace, and *(ii)* to determine the timescales at which it is appropriate to consider symmetry. Our results are shown in Figure 4.

Regardless of the time-scale over which we measure $\mathcal{S}$, the vast majority of hosts exhibit strongly symmetric traffic ($|\mathcal{S}| \leq 2.0$). The left-hand tails depict hosts where a considerably larger number of packets were received than transmitted, while the right-hand tails show the opposite. The smallest time-scale (one second) most clearly separates symmetric and asymmetric hosts, but requires much more state be tracked. Note that the plots for one hour and 24 hour windows completely overlap, illustrating that no self-similarity is observed in terms of traffic symmetry.

4.2 Host-Pair Packet Symmetry

Next we decided to investigate the level of symmetry observed between the unique pairs of hosts in the trace. We carried this out for the approximately 320,000 pairs in which both source and destination send packets and use a time-scale of 1 minute to calculate $\mathcal{S}$.

Figure 5 shows the cumulative distribution of $\mathcal{S}$ for all host pairs that exchanged packets in both directions during our observation period. Almost all host pairs maintain extremely strong symmetry in their communications ($|\mathcal{S}| \leq 1.0$), while very few are significantly skewed towards the receiving side (bottom 1%) or the transmitting side (top 3%).

We also measured $\mathcal{S}$ for a further 6.8 million host pairs in which only the source sends any packets. This clearly undesired traffic demonstrated symmetry values ranging from 0.69–10.5, although with 99.9% less than 2.0.

4.3 Flow Symmetry

To investigate the symmetry of traffic within individual flows, we chose to examine separately the sets of TCP and UDP flows within the trace. For increased precision, we calculate symmetry values every second.

Figure 6 shows the cumulative distribution of the maximum value of $\mathcal{S}$ for the TCP flows in the trace. The use of acknowledgment packets in TCP imposes a degree of symmetry on all flows in the trace; virtually all TCP flows exhibit asymmetry ≤ 1.5—a ratio of about 4.5 packets to one. Examining the outlying TCP flows reveals a small number of misbehaving (or at least irregularly behaving) flows, which we are currently investigating.

Considering UDP flows, Figure 7 shows a much broader

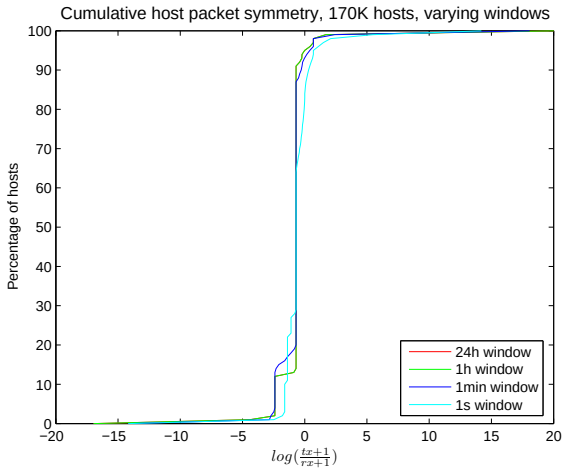

Figure 4: Host Symmetry

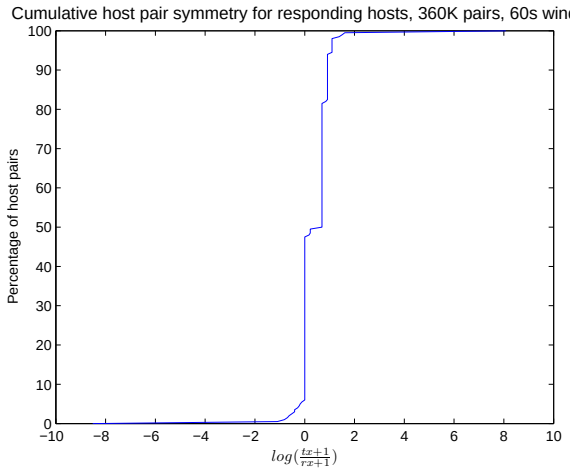

Figure 5: Host-Pair Symmetry ($rx > 0$)

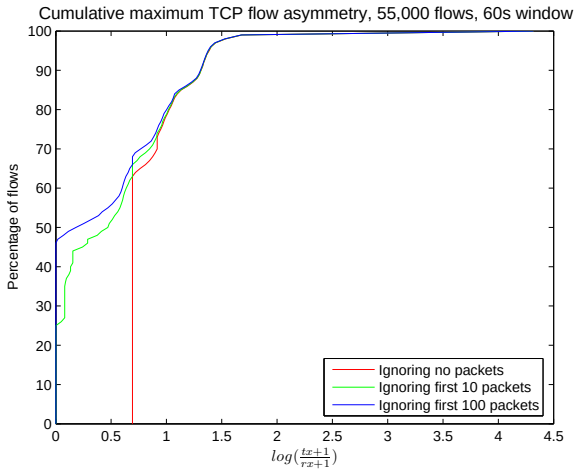

Figure 6: Maximum per-flow asymmetry (TCP flows of length > 100)

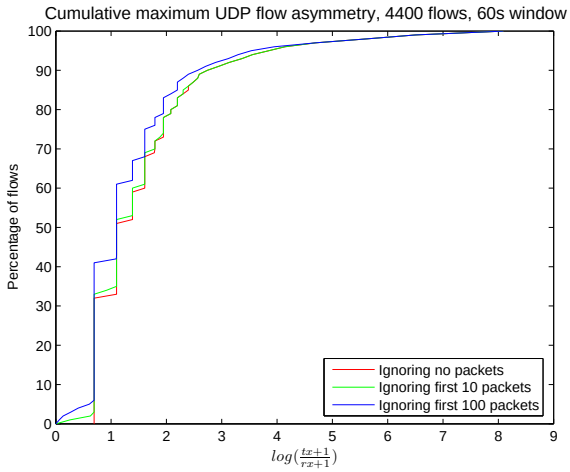

Figure 7: Maximum per-flow asymmetry (UDP flows of length > 100)

range of symmetry values. On further examination of the outlying UDP flows we find a great deal of misconfigured DNS traffic, and a considerable number of malicious flows; all of these packets are clearly supposed to be subjected to our shaper.

In both sets of flow measurements we examine the effect of ignoring packets at the beginning of the flow to reduce any transient asymmetry present with low packet counts. Both of the flow-granularity CDFs use only flows from the trace with an excess of 100 packets, and demonstrate that a smoothing effect may be obtained in this manner.

In summary, our analysis shows that packet-level symmetry shows good promise as a classifier between well-behaved and abusive traffic. Furthermore, monitoring per-flow symmetry did not indicate fundamentally different properties than did host-pair granularity, leading us to believe that the latter is a good starting point for future investigations.

5 Discussion

We next consider possible evasions, discuss deployment options, and argue for making symmetry a *required* component of protocol design.

5.1 Evasive Manoeuvres

We first consider the steps attackers might take to fool our mechanism into incorrectly allowing a node to keep sending substantial amounts of undesired traffic. The degree to which we can avoid this depends on the amount of state we use and whether or not we assume spoofable source addresses. In the following, we assume host-pair symmetry tracking. We believe this amount of state to be easily manageable, particularly given deployment close to the edge, while allowing our enforcement to be protocol agnostic.

Without help from the outside, individual hosts can only dilute their asymmetric traffic by ensuring a large fraction of symmetric traffic. This is clearly a minor concern. A first distributed strategy is to “fly under the radar” individ-

ually, that is, to keep the abusive traffic to a small amount per node, but use substantial amounts of nodes for the attack, i.e., to leverage a botnet. Attackers could, for instance, use a pulsing attack where all nodes attempt to blast away as much traffic as possible, then fall quiet and just as the throttling mechanism permits new transmissions, blast away again. Our answer is that a suitably sensitive delay mechanism would only permit such blasts for a very brief period (recall how in Figure 1 the UDP flood was muted very quickly).

Another strategy is collusion: during a DDoS attack, the attacker uses the nodes of a botnet to send spoofed cross-traffic amongst its members in order to fool the members' symmetry monitors into thinking that the victim is returning substantial amounts of packets. The feasibility of this approach depends on the following:

- Source address spoofing. As network ingress filtering [9] becomes more pervasive, this will become increasingly hard. Note that our symmetry mechanism, if widely deployed, would be an ideal opportunity to enforce widespread deployment of ingress filtering.
- Randomization of IP ID values by the victim. Without this feature, a significant number of colluding nodes have to be informed about the ID value, estimate the IP IDs of the forged packets, and do this quickly enough so forged packets are labelled accurately *and* arrive in a reasonable sequence.
- TTL estimation. Every bot needs to discover by itself the correct TTL value that matches the TTL of the actual victim's reply packets as they arrive at the attacking bot that is colluded with.

We feel that while better defence against spoofing-based collusion is conceivable, the IP ID and TTL combination will be difficult enough to overcome for the period until our mechanism is deployed widely. Once that is the case, pervasive symmetry enforcement and ingress filtering allow the attacker at best fragile layered asymmetry exploitation to achieve limited amounts of gain in abusive traffic. Furthermore, our approach to reducing botnet effectiveness is aided by an increasing array of other mechanisms highly compatible with ours, such as rigorous checking of reverse path forwarding at the ingress point [10].

5.2 Deployment Considerations

We envision three basic deployment strategies. First, virtualized co-location facilities can use virtual machine monitors such as Xen [11] to provide administrators with a control mechanism that ensures their sites cannot be abused for attacks. Similar concerns have recently arisen in context of research networks such as PlanetLab and we are interested in exploring deployment opportunities in these environments as our implementation matures. Second, deployment can occur in the form of "smart NICs"— network interfaces which may be programmed with advanced functionality [12]. Third, access networks provide a single monitoring point with full access to the networks' traffic, for example at edge routers. In all of these scenarios, deployment is beyond the reach of potentially compromised application/OS software while still remaining sufficiently close to the edge of the network that packet provenance is not questioned.

5.3 Mandating Symmetric Protocols

Our experiments have already stressed a new rule for good protocol design: symmetric protocols are easier to control. Not only do they allow restricting the sender to a certain threshold asymmetry, they also allow the receiver to implicitly signal end-of-interest simply by opting not to respond. At the same time, requiring symmetry also prevents attacks that abuse granularity incongruences in protocol control mechanisms [13]. We believe an analysis of existing protocols from a symmetry perspective could prove highly fruitful in evolving the approach outlined in this paper. In extremis, we might imagine moving symmetry-based enforcement towards a much stronger (e.g. $|S| \simeq 0.0$) threshold. Enforcing drastically lower levels of asymmetry would likely reduce the efficiency of some existing protocols (thus acting as an impetus to upgrade), while concurrently providing an even clearer distinction between good and bad traffic.

6 Related Work

The study of malicious traffic has been of such significant interest that a thorough list of citations would easily fill the length of this paper. Generally speaking though, we believe that the approach described here is the first to argue for a simple, incremental change to the Internet based on enforcing the notion that symmetric traffic is good traffic.

Our approach is a proactive, constraint-based change to the network that intends to prevent it from getting into a bad state in the first place. In contrast, many existing approaches have been much more reactive in nature, involving a receiver signalling that they are under attack. We feel that these approaches address the symptom rather than the disease; responding to an active attack is incredibly difficult often requiring architectural support to identify the source of the attack [4, 5, 6] and support from routers to offload filtering closer to the source [14, 15].

A significant additional flaw in these approaches is that damage is already done before countermeasures can be taken: intermediate network links may become saturated, resulting in collateral damage in accessing neighbouring receivers who are not directly under attack. This reasoning has resulted in several recent proposals for considerably more drastic architectural changes, often making connection signalling explicit [1, 2, 3]. We believe that these proposals are overly complex, and would be considerably more difficult to deploy than ours.

Several other efforts bear individual similarities to our

work. Microsoft's recently produced—and withdrawn—TCP connection limiting for SP2 [16] placed a strict limit on the rate at which new TCP connections could be opened. MULTOPS [17] described a router design that used a similar approach to us; their work is limited to DDoS limitation and TCP traffic, and focuses considerably more on router data structures than on traffic analysis.

Perhaps most closely related are the D-WARD proposal [18] and the MANAnet Reverse Firewall [19]. Both of these propose throttling DDoS traffic close to the source, although they focus on byte- rather than packet-level metrics, and use more involved algorithms requiring additional state and computation. Moreover, our approach does not require access to the contents of packet payloads.

7 Conclusion and Future Work

This work argues that packet symmetry should become a fundamental design principle for the Internet. To support our argument, we have introduced a simple symmetry metric, illustrated a symmetry enforcement prototype, justified it using trace-driven analysis, and described an incremental deployment path that benefits the Internet's major hosting sites immediately.

Our work poses a number of open questions that we intend to address in future work. First, what is the optimization of the state vs. accuracy tradeoff? It is not yet clear what granularity of traffic our final implementation will need to function at. As we intend a deployment which will function at hundreds of megabits and serve a thousand hosts, we are aware that the state overhead will need to be carefully minimized.

Second, are there benign applications that we have missed? Our analysis is based on a 24 hour traffic at a major non-university research institution. We are fairly confident that virtually all predominant applications, both benign and malicious, are represented. We have not yet found any applications that would be adversely effected by symmetry-based classification. The solitary example we are aware of is that of unacknowledged broadcast and multicast transmission, which are easy to treat specially and could be made conformant by low-rate acknowledgement streams.

Acknowledgements

The authors would like to thank Mark Allman, Vern Paxson, Chema Gonzales, Juan Caballero, and Atanu Gosh for many helpful comments on this work. We also feel indebted to Michael Dales for demonstrating how easily the approach described in this paper can be implemented in hardware (a naive implementation in less than 200 lines of VHDL), and to Andrew Moore for providing the traces for the analysis.

References

[1] D. Adkins, K. Lakshminarayanan, A. Perrig, and I. Stoica. Taming IP Packet Flooding Attacks. In *Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[2] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *Proceedings of the Second Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[3] M. Handley and A. Greenhalgh. Steps Towards a DoS-resistant Internet Architecture. In *Proceedings of the Second ACM SIGCOMM Workshop on Future Directions in Network Architecture (FDNA-04)*, pages 49–56, 2004.

[4] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Network support for ip traceback. *IEEE/ACM Transactions on Networking (TON)*, 9(3):226–237, Jun. 2001.

[5] A. C. Snoeren, C. Partridge, C. E. Jones L.A. Sanchez, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet IP traceback. *IEEE/ACM Transactions on Networking (TON)*, 10:721–734, Dec. 2002.

[6] A. Yaar, A. Perrig, and D. Song. Pi: A path identification mechanism to defend against ddos attacks. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pages 1–15, 2003.

[7] A. Moore and K. Papagiannaki. Toward the Accurate Identification of Network Applications. In *Proceedings of Sixth Passive and Active Measurement Workshop (PAM 2005), Boston, MA*, March 2005.

[8] A. Moore, J. Hall, E. Harris, C. Kreibich, and I. Pratt. Architecture of a Network Monitor. In *Proceedings of Fourth Passive and Active Measurement Workshop (PAM 2003), Boston, MA*, April 2003.

[9] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827, IETF, May 2000.

[10] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704, IETF, March 2004.

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating Systems Principles (SOSP19)*, pages 164–177. ACM Press, 2003.

[12] I. Pratt and K. Fraser. Arsenic: A user-accessible gigabit ethernet interface. In *Proc. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM-01)*, pages 67–76, Los Alamitos, CA, USA, April 22–26 2001. IEEE Computer Society.

[13] Stefan Savage, Neal Cardwell, David Wetherall, and Tom Anderson. Tcp congestion control with a misbehaving receiver. *SIGCOMM Comput. Commun. Rev.*, 29(5):71–78, October 1999.

[14] R. Mahajan, S. Bellovin, S. Floyd, J. Vern, and P. Scott. Controlling high bandwidth aggregates in the network, 2001.

[15] K. Argyraki and D. Cheriton. Active internet traffic filtering: Real-time response to denial of service attacks. 2005.

[16] (permission pending), 2005. Personal Communication.

[17] T.M. Gil and M. Poleto. MULTOPS: a Datastructure for Bandwidth Attack Detection. In *Proceedings of the 10th Usenix Security Symposium*, Aug 2001.

[18] J. Mirkovic, G. Prier, and P. Reiher. Attacking DDoS at the Source. In *Proceedings of 10th IEEE International Conference on Network Protocols*, Nov 2002.

[19] CS3, Inc. MANAnet Reverse Firewall: Fighting DDoS Attacks at their Origins. `http://www.cs3-inc.com/ps_rfw.html`.