Stress Testing Traffic to Infer Its Legitimacy

Nick Duffield and Balachander Krishnamurthy
AT&T Labs–Research, 180 Park Avenue, Florham Park, New Jersey, 07932, USA
{duffield,bala}@research.att.com

Abstract

Adaptation in the face of performance degradation is the hallmark of well-behaved network traffic. For sufficiently robust applications, we propose distinguishing good from bad traffic on the basis of the response to artificial performance impairments. We explain the basic requirements for our scheme, and show how it could be generically applied at different levels in the protocol stack.

1 Introduction

This paper proposes a new measurement-based methodology for detecting badly behaving network entities. The methodology can potentially be applied across a range of protocol levels. The key idea is to stress test an entity by impairing its network performance, and then to observe the response of the entity to the impairment. Our work is primarily oriented closer to the edge rather than the middle of the network.

The usefulness of the method rests on two assumptions:

- **Differentiation:** The response to impairment is different for bad entities than good entities, and can hence be used to classify entities as good or bad.
- **Recovery:** good entities can recover from impairment in the sense that the performance degradation they suffer remains within acceptable levels.

In practice, these two assumptions are coupled. Suppose we impair the performance of a protocol that is used by an application. We require the underlying protocol to be more sensitive and responsive to impairment that the application, i.e., it must adapt to the impairment before the application suffers an unacceptable performance degradation. Impairment must be sufficiently short-lived to be interpreted as a transient, forcing retry, leading to recovery. The nature of this response is used to distinguish bad from good.

For some applications, e.g., online gaming and others that are highly sensitive to loss and delay, *no* impairment is tolerable. We assume that these applications can be identified (e.g. on the basis of TCP/UDP port numbers) and excluded from our scheme. More generally, the impairments induced by stress testing must not cause performance degradation to exceed acceptable limits, e.g., as specified in a Service Level Agreement (SLA).

2 Impairment and Responses

We now describe the broad characteristics of impairments, propose some schemes for good/bad classification, and discuss the costs of misclassification.

2.1 Impairment Characteristics

In many cases, applications will experience performance impairments even in the absence of stress testing, for example due to packet loss, delay, network rerouting or resource contention at endpoints. We call these impairments *ambient*. We assume that applications are well-designed in the sense that they a reasonably good at adapting to or otherwise withstanding ambient impairments. If this assumption holds, then it makes sense for impairments introduced by stress testing to conform as closely as possible to the characteristics of the ambient impairments. For then applications need only recover from impairments of the type (even if not the intensity) of those that they are known to adapt to. Another reason for stress impairments from to conform to ambiance is to make it harder for an adversarial bad application to detect that it is being stress tested.

This begs the question of the extent to which ambient impairments can be characterized and then reproduced in stress testing. We will discuss this further for specific applications in Section 3. Here we enumerate some general parameters for impairment. In our model, the stress testing takes the form of a sequence of individual stress events, characterized by the following parameters:

- *Frequency:* of stress events during stress testing
- *Event duration:* the duration of an single stress event
- *Stress duration:* the duration of a set of stress events. Note that the stress duration may itself be adaptive in the sense that stress testing is terminated once sufficient information has been extracted from responses for classification, or if it is decided that no such classification can be made.
- *Granularity:* at which stress take place. For example, a stress test might target all packets from an IP address, or a subnet, or using a particular UDP/TCP port.

How should these parameters be determined? The aggressiveness of the stress test should be commensurate with the perceived threat. The frequency of bad traffic may be bootstrapped by learning directly from the identification scheme that is used in the stress test, although their will be a transient period in which such estimates are unreliable. One way to seed an initial the threat level is to monitor the intensity of bad traffic taking place, e.g., attacks on "dark" address space as seen in a honeypot [9]. A generic scheme to set parameters for the stress test is described in Section 2.3 below.

2.2 Impairment and Control

We use the following three outcome model to describe the response of an entity to a stress event:

- *Cease:* the entity ceases all activity. This may be due to a bad entity having turned its attention elsewhere, or any entity suffering an unrecoverable error as a result of impairment. Whatever the reason, the current stress test of the entity is terminated.
- *Evolve to Good:* adaptation to impairment in the manner expected of a good entity. The current stress test of the entity is terminated.
- *Evolve to Bad:* any other result, for example, not responding to impairment in any manner. The entity is flagged as "suspicious". Stress testing may continue.

How should the outcome of one or several stress events be interpreted? The greater the number of stress events, the greater the certainty of the classification based upon the outcomes. This motivates a cautious approach in which only multiple suspicious outcomes would result in an entity being designated bad, in which case some further action, e.g., blocking, is taken. This may be accomplished in a number of ways; here in one:

- *Fixed horizon classification:* a fixed number n of stress events is generated in sequence. An entity is classified as bad if the number of suspicious outcomes exceeds some level $m \leq n$. The history of outcomes is discarded for each new stress test. probes.

Impairment may also be used as means of control; in this case the intensity of impairments (as determined by frequency and/or duration of stress events) can be increased in response to the level of suspicion. At the same time, it is desirable not to penalize an entity unduly for suspicious behavior far in the past. Thus one penalizes bursts of suspicion, while admitting ambient low level of false positives. This motivates:

- *Queue-based classification:* Suspicion flags are enqueued in an infinite buffer which drains at some rate. The impairment intensity is a function of the buffer occupancy.

2.3 Costs of Impairment

We use the term *cost* to denote the potential undesirable impact of the impairment method on applications. We identify two types of cost:

- The **impairment cost** represents the performance degradation experienced by good applications during impairments, and during their response to impairments. This cost includes, for example, the impact of packet loss or delay due to impairment on an application's performance.
- The **identification costs** are those of actions taken on the basis of the identification of applications as good or bad. Examples of such costs include the the false positive rate (the frequency with which good applications are misidentified as bad) and the false negative rate (the frequency with which bad applications are misidentified as good). Impairment may be combined with other classification methods. In this case, the relevant identification costs are those of the composite classifier.

For a number of reasons, both impairment and identification costs may vary with time. As noted, a stress test may adapt to the responses of the target, either to increase the effectiveness of classification, or to control entities classified as bad. Adaptation takes the form of changing the intensity of the impairment, or possibly removing it altogether. This results in a concomitant changes in the false negative and positive rates.

3 Applications

We now examine the key set of protocols on a case by case basis. The main issues are: the possible responses as a result of impairment (apart from normal response and no response), how the responses may be interpreted at the client side, are they recoverable, and if so how (where will the adaptation be seen if at all). Impairment and monitoring has to take place at the same protocol level to facilitate tracking of the reaction. Our guiding principle is that our scheme becomes difficult to use at a given protocol level if there are a variety of adaptation mechanisms possible or used by traffic using that protocol that cannot be distinguished. Differentiation typically requires moving to a higher layer.

Note that in some protocols/layers impairments already being used or subsumed by other mechanisms. Where appropriate we mention them. For example, in BGP the notion of impairment is not so crucial as the peers exchanging routes and/or traffic have prearranged thresholds.

3.1 UDP and its Applications

UDP does not provide any adaptation mechanisms in itself. However, higher level applications and protocols may select UDP as their transport, but implement their own adaptation to impairment. In some cases the adaptation to be evident at the transport layer, involving reduction of the sending rate (to avoid congestion) and the repetition of packets (for reliability). The precise details of the adaptation will depend on the higher level protocol or application in question. The UDP port number can be useful in identifying the application. More subtle adaptations may only be visible in the transport payload, and again their location and interpretation would rely on accurate identification of the application in question.

For a specific example, we consider the Realtime Transport Protocol RTP [14] running over UDP. Within RTP, The RTP control protocol (RTCP) provides feedback on the quality of data distribution. Under impairment due to stress testing, receivers will notify senders of reduced quality via RTCP. A good sender is expected to modify its transmissions accordingly. However, the nature may appear quite subtle at the network level , e.g., reducing the number of layers sent from a layered encoding of audio or video. The precise nature of this adaptation is determined according to the policy of the end application.

3.2 TCP and Other Congestion Avoiding Transport Protocols

TCP is equipped with congestion avoidance mechanisms: a well-behaved TCP adapts to congestion by decreasing its congestion window before growing it again to explore the available bandwidth. The TCP sender detects congestion when acknowledgements are not forthcoming from the receiver, due to loss and/or delay of packets in transit. Thus a TCP connection should both adapt to impairment, and also recover from it. Stress testing of TCP is accomplished by dropping and/or delaying packets, and observing the connection's response. Impairment must not be so intense as to seriously degrade throughput. A connection which either adapts either too slowly, or not at all, is regarded as bad. Thus, depending on the thresholds for classification. TCP sender that is too aggressive may be regarded as bad.

We give more detail on how the framework might be implemented for impairment by packet loss. We require to be able to monitor and possibly impair both directions of a TCP connection. One way to achieve this is to insert the desired functionality below the TCP layer in a receiving host. Impairment is achieved by selectively dropping packets incoming from a sender. Using the terminology from Section 2, a normal TCP connection is expected to "evolve to good" as it adapts to dropped packets.

A number of approaches to passively characterizing TCP connections have been proposed; see [4, 5, 19]. One aim of this work has is to compare measured characteristics with theoretical baselines. Deviations from these baselines would then be used as triggers for routers to impose limits, e.g. by restricting the bandwidth available to a connection in order to restore a fair distribution of bandwidth. Our aim is somewhat different: to identify and penalize bad connections in advance of their impact on other traffic. [4] proposed three categories: flows that are not TCP friendly, flows that use a disproportionate amount of bandwidth, and flows that are unresponsive to packet loss. The last case matches best with our framework; [4] mentions the idea of introducing packet loss impairments, and observing whether reduction in throughput (if any) conforms to expectations.

A more detailed yardstick against which to measure the sender's response to impairment is an estimate of the sender's congestion window obtained by monitoring the receiver-to-sender ACKs. This has been proposed in [5], which uses the ACKs to drive transitions in a finite state machine (FSM) that represents the sender. State transitions due to sender timeouts may be inferred by monitoring sender-to-receiver retransmissions. Parallel FSMs may be run for different flavors of TCP. Losses occurring between the monitoring point and the sender lead to estimation uncertainties, although the impact can be detected and corrected for to some degree.

We outline how classification of bad senders might proceed if using the congestion window estimation scheme. A sender is flagged as suspicious if, it sends a packet that would not be allowed by TCP under the current estimate of the senders congestion window. The sender is flagged as bad if it sends multiple suspicious packets, e.g., as determined by one of the classification schemes in Section 2.2. In principle, this classification could be made of any connection, whether impaired or not. The advantage of stress testing is that by active soliciting a response from the sender, it enables the identification of bad senders in advance of circumstances where their bad behavior may be problematic. In this example stress testing helps identify overaggressive TCP senders in times of low network congestion, rather than in congestion periods in which they could have a disproportionate impact on other connections. This is somewhat different to the approach of [4], which proposed characterization of higher rate flows during periods of congestion. Once a sender is classified as bad, some action may be taken against it (e.g. dropping, blocking, or otherwise deprioritizing) at the monitoring point, or by reconfiguring routers in the traffic's path.

We propose to evaluate this approach in future work. Initial evaluations will use controlled TCP-like senders that can be configured to act in good or bad fashion. The existing active TCP inference tool `tbit` [13] can likely be modified for this purpose.

Stress testing is intended for connections that are already established. On the other hand dropping packets at the start of a connection, specifically the SYN and SYN-ACK packets used in the three-way handshake that establishes the connection, should be used sparingly or not at all, since the performance impact of preventing connections establish altogether will be far greater.

Although the operational details may differ, a similar approach to stress testing can be adopted with other congestion avoiding transport protocols, including the Stream Control Transmission Protocol (SCTP) [16] and the Datagram Congestion Control Protocol (DCCP) [7].

Although we have described an example in which monitoring and impairment takes place at a connection endpoint, the same could be accomplished in the middle of the network. (The methods of [5] were originally designed for this context). Here, keeping track of the identification status of potentially a huge number of connections is a challenging. But since their are only limited number of potential states in the classification schemes of Section 2.2, one may use compression schemes such as Bloom filters [1] in order to reduced the space needed. For example, in the fixed horizon classification scheme one may maintain a Bloom filter each number of suspicious outcomes, and store keys of connections in it. The current state is then yielded by the highest level Bloom filter that declare a match on the connection key. However, Bloom filters are subject to false positives, which increases the identification costs.

3.3 DNS

A disproportionate amount of DNS traffic consists of UDP exchanges involving port 53; apart from zone transfers there is very little TCP traffic. Traditionally DNS clients are configured to send recursive queries as their resolver libraries often do not follow referrals. Local DNS servers can handle iterative referrals that might come back from authoritative DNS servers. One weakness with impairment in DNS is that many clients may ask multiple name servers in parallel for non-authoritative queries. Authoritative DNS servers however can impair requests by referring the request to non-existent servers to see if the query comes back. Root servers are ill equipped to participate given how busy they are with the already large fraction of illegitimate, albeit non-attack, queries [18] that they receive.

3.4 SMTP

The Simple Mail Transfer Protocol has reasonable potential for impairment trials. Although email has the reputation for being delivered instantaneously, the dependence on this has significantly eroded due to the popularity of instant messaging and the rise of email spam. A non-trivial number of email messages are delayed for a variety of reasons and the retry mechanism built into the application (with retrials carried out over several days) ensures that the mail will be eventually delivered.

Our assumption here is that spammers are not likely to retry sending the mail as it would require human intervention. Robots would have to be able to parse the reply and separate simple bounce messages from vacation programs and impairment triggered responses. However, depending on the importance of the mail, the human (non-spammer) sender may retransmit the mail. Each such retransmission dramatically lowers the probability of the sender being a spammer. Many sites maintain a whitelist (good senders), blacklist (bad senders), and graylist (as yet unclassified senders). Retransmitters could be moved from the graylist to the whitelist. The related step of moving non-retransmitters to a blacklist is however trickier as is the problem of dealing with mailing lists.

A serendipitous benefit for users who have been selected for impairment is that they faced a one-time delay but future communications are likely to improve. A spammer trying to subvert the mechanism would have to retransmit a very large number of messages in trying to game this mechanism with a low probability of guaranteed success.

3.5 HTTP

At the HTTP layer, redirection would involve a 3xx level response (redirection class response code such as `307 Temporary Redirect` but to a blank page. Alternately the server could send back a `408 Request Timeout` or a `503 Service Unavailable` along with a `Retry-After` header indicating the number of seconds after which the client should retry the request. Here, this value should be as low as possible; say 1 second. The assumption again is that a user may retry but an attack program is not likely to react the same way. With the Web server under our control, it is possible to monitor the frequency of access and retries as input to further impairment actions. Interestingly, the expected adaptation on the part of the user is to retry whereas in TCP the expectation is the opposite: backing off.

We benefit due to the additional and more detailed semantic information available at the Web server. For example, it is possible to distinguish between normal user requests, and those that emanate from spiders. Thus, a Web server operating under impairment rules, can make judicious choices of URLs whose access trigger impairment guided by its access patterns and popularity.

3.6 P2P

Three broad categories of P2P protocols have emerged: the original central index model of Napster, the flooding of neighbors with separation of search and download phases model exemplified by Gnutella and KaZaa, and the most popular tracker process and segmented downloading model of BitTorrent. There are at least two schools of thought: one that presumes virtually all the content exchanged on P2P networks are of dubious legality and the other that views that there is a steadily increasing share of significant amounts of legal content being exchanged. For example, Linux Kernel releases are now routinely copied via BitTorrent.

One impairment technique already prevalent is content pollution: insertion of white noise in audio files or unexpected content in large video files, to frustrate the illegal downloaders [10]. Another prevalent technique–that of choking or throttling connections–is to reduce freeloading and to look for other peers who might be able to share wanted content faster. We consider the latter technique here.

Elimination of freeriding that began in eMule and was extended in BitTorrent revolves around the use of tit-for-tat mechanisms by giving poor service to nodes that do not dedicate enough of their bandwidth to uploading. Here impairment is done by deliberately scheduling poor sharers at the rear end of the queue. Another impairment technique available in protocols like BitTorrent is an *early choke mechanism* whereby nodes who are constantly looking for better connected nodes can drop one of the existing (poor) connections. While this is done for a selfish purpose, it can address concerns about nodes that may pretend to have poor connectivity. However, the false positives will be detrimental to the choked node.

The absence of both local history (between a pair of nodes) and global history (across nodes) leads to false positives (well behaved nodes are mistakenly impaired) and false negatives (nodes that don't share enough receive good treatment). Similar to the TCP case where we argued for trying impairments on long lasting connections only, we believe such impairments should only be tried when there is enough information about the other nodes.

4 Relation to Existing Approaches

Variants to our proposed impairment notion have been proposed both in the middle of the network and at the edge in the past. In the TCP arena, the notion of penalty box has been advanced, as a way of restricting bandwidth usage by misbehaving flows [4]. At user-level applications such as Email it is possible to challenge email senders with a simple puzzle to be solved in order to distinguish human senders from programs. It is important to note a key distinction between proposals made in the past and ours: the primary issue was fairness in the past with a different quality of service offered to misbehaving flows arising out of, presumably accidental, misconfigurations. We are more interested in identifying malicious and deliberate senders of unwanted traffic. Once an email puzzle is solved the sender is permanently classified as good by being added to a whitelist thus leading to potential abuse by forging addresses of whitelist members.

Honeypots [9, 15], a resource whose value lies in its unauthorized use, advertise unused address space and examine traffic that arrive there. Honeypots can help identify suspicious IP addresses [17]. Some honeypots listen passively, while others actually interact with the traffic by responding to connection set up attempts. At the other extreme, there are honeypots that can emulate a kernel keeping the attacker busy with fake responses. In the past, tarpits [8] have been deployed to actually waste resources of suspicious attack sources. However, the key difference is that virtually all traffic arriving at the dark address space is known to be unwanted. In the domain of electronic mail, the MAPS [11] black list of abusers of email allowed any system administrator to drop all email coming from these domains. Again, the impairment here is after the identification of the (potentially) offensive source.

The throttling of connections suspected not to share an equal fraction of their bandwidth for uploading in P2P networks, is clearly a form of impairment. However, such tit-for-tat mechanisms is on a one-to-one basis and is based on immediate recent history of transactions. Such a mechanism requires evaluation of all of the node's neighbors. The impairment can be temporary as choked connections can be unchoked later on a case-by-case basis. The granularity of impairment is at the connection level rather than on packets. Even the semantic queuing of suspected poor sharers by placing them towards the tail end of the queue can lead to starvation of some nodes indicating absence of fine-grained control.

5 Strengths and Limitations

In this section we discuss strengths and limitations of stress testing: the need to control impairment cost, the scope for countermeasures, and the trade-offs in using application level semantics for identification.

5.1 Keeping Impairment Costs Acceptable

Impairment costs must not be so large as to make stress testing unacceptable for good network users. Other approaches that have been proposed include doing limited flooding of links that are suspected to be bearing attack traffic [2] and the more recent proposal to attempt to generate challenges to suspected email senders [12]. In the former case there is a risk of doing more damage than warranted or beneficial while in the latter zombie machines generating email may get challenges. We focus on keeping impairment within an SLA or similar limits.

The total impairment experienced by an entity has two components: the ambient impairment and the impairment due to stress testing. The total costs must be kept within acceptable limits. This requires two things.

Firstly, the acceptable limit must be known. This may be expressed in terms of SLAs between a service provider and its customers, that governing the allowable performance degradation and penalties for deviations therefrom. However such limits may need to be applied conservatively, since customers are commonly responsive to degradations beyond an "effective" SLA corresponding to the level of service that they usually receive, which may be better than the worst degradation allowed under the SLA.

Secondly, the level of ambient impairment must be known. Depending on the application, this may be known from application level statistics, e.g. as maintained by Web servers, or from network measurements. In the latter case, one challenge is to characterize ambient impairments at the same level of granularity as the stress test itself. Although commonly reported loss statistics, such as those reported via SNMP, are aggregated at the link level, a more recent method, Trajectory Sampling [3], allows the estimation of loss rate (and sometimes packet delays) at the level of individual hosts and links.

5.2 Scope for Countermeasures

A strength of stress testing is that the scope for countermeasures from a bed entity is limited. Firstly, the impairments in a well-designed stress test are not distinguishable from ambient impairments, and thus it is difficult to determine that stress testing is taking place. This likely entails using a full spectrum of likely impairments (e.g. including both loss and latency) in a suitably randomized manner that leaves no signature. Secondly, the method is potentially ubiquitous, making reverse blacklisting by bad entities harder. Thirdly, response to impairment in an aggressive manner makes it more likely that the entity will be flagged as suspicious or even identified as bad.

5.3 Application Layer Semantics

In principle a detailed understanding of application semantics may be used to construct detailed models of expected user behavior, and departures therefrom detected. However, this approach becomes more problematic the further up the protocol stack one goes. Applications may be deployed in multiple hosts and in different variants; thus it is necessary to either modify applications or write appropriate dynamically linked libraries. At the network layer, with the small number of transport protocols, it is easier to implement impairments. The trade-off is that with knowledge of application-specification semantics it is easier to construct narrow and specific impairments to identify potential attackers. For example, a Web server could track access patterns on its site and selectively redirect requests for rarely requested URLs when the number of requests exceed a threshold or when they come from BGP prefixes never seen before [6]. However, this requires state maintenance about URL access patterns and tuning when the site changes.

6 Impairment factors

Below we list various factors to be kept in mind while considering impairment. They include where impairment should be done, directionality of traffic, for how long, and how meaningful thresholds on false positives and false negatives can be generated.

Impairment is done most readily at the receiving network's ingress point or at the application host. It does not involve any resources in the middle of the network.

Should impairment be done on incoming traffic only or can we consider outgoing traffic too? Depending on the application it may well make sense to impair outgoing traffic. For example, if there is a suspected virus loose behind a network triggering large volumes of email or outbound portscans, it might be reasonable to selectively drop some transactions as a pre-throttling measure. An added benefit in dealing with outgoing traffic is that there is a better prior history of patterns to make informed and selective impairments.

Depending on the infrastructure being protected, the cost of false negatives and false positives, the duration and nature of impairment will vary within applications. A DNS attack against a typical site is a lot less problematic than against a CDN infrastructure which depends on DNS for its entire service. In such instances, cost of a false negative is high and it may pay to be more aggressive. Thus, where there is a potential of amplification of attacks, frequency of impairment attempts can go up.

In some cases, the parameters of the stress test can be set based on a cost analysis related to that in [20]. Suppose that the false positive f_+ and false negative f_- rate for identification are known and a function of test parameters ϕ. This knowledge can come from analysis of a model of the test, or from empirical observations, or both. Suppose a proportion p of the total traffic is believed to be bad traffic. p may be a current estimate based in the stress testing itself, or as determined by independent measurements. Then the parameters ϕ are chosen so as to minimize the total cost $C(\phi) = pf_-(\phi) + (1-p)f_+(\phi)$. Note that the test parameters automatically adapt to changes in the measured or expected value of p.

7 Summary

We have outlined a new technique, impairment via stress testing, to distinguish good and bad traffic. The technique is low cost, tailored to different applicable layers, and thresholdable to ensure no service level agreements are violated. The natural concerns of false positives and negatives are discussed via tunable parameters governing the stress causing events. The technique is more natural at the transport layer than others although we can take advantage of the broader leeway available in SMTP, HTTP, and P2P applications. A key advantage of stress tests is that attackers cannot easily detect them and will likely become more visible by trying to counter the measure.

Acknowledgments

We would like to thank the anonymous referees for their useful feedback and pointers to earlier related work.

References

[1] BRODER, A., AND MITZENMACHER, M. Network applications of bloom filters: A survey. In *Proc. Allerton Conference* (Monticello, IL, October 2002).

[2] BURCH, H., AND CHESWICK, B. Tracing anonymous packets to their approximate source. In *Proceedings of Usenix LISA conference* (2000).

[3] DUFFIELD, N., AND GROSSGLAUSER, M. Trajectory sampling for direct traffic observation. *IEEE/ACM Transactions on Networking 9*, 3 (June 2001), 280–292.

[4] FLOYD, S., AND FALL, K. Promoting the use of end-to-end congestion control in the internet. *IEEE/ACM Transactions on Networking* (August 1999).

[5] JAISWAL, S., IANNACCONE, G., DIOT, C., KUROSE, J., AND TOWSLEY, D. Inferring tcp connection characteristics through passive measurements. In *Proceedings of IEEE INFOCOM* (March 2004).

[6] JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites. In *WWW* (MAY 2002).

[7] KOHLER, E., HANDLEY, M., AND FLOYD, S. Datagram Congestion Control Protocol (DCCP). Internet Draft, March 2005.

[8] Hackbusters - Homepage. `http://hackbusters.net`.

[9] LANCE SPITZNER. Honeypots: Definitions and Value of Honeypots. `http://www.tracking-hackers.com/papers/honeypots.html`.

[10] LIANG, J., KUMAR, R., XI, Y., AND ROSS, K. Pollution in p2p file sharing systems. In *IEEE Infocom* (2005).

[11] The maps realtime blackhole list. `http://mail-abuse.org/rbl/`.

[12] NELSON, M. Fairuce, March 2005. `http://www.alphaworks.ibm.com/tech/fairuce`.

[13] PADHYE, J., AND FLOYD, S. On Inferring TCP Behavior. In *ACM SIGCOMM* (2001), pp. 287–298.

[14] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, July 2003.

[15] SHAHEEM MOTLEKAR. Honeypots: Frequently Asked Questions. `http://www.tracking-hackers.com/misc/faq.html`.

[16] STEWART, R., XIE, Q., MORNEAULT, K., SHARP, C., SCHWARZBAUER, H., TAYLOR, T., RYTINA, I., KALLA, M., ZHANG, L., AND PAXSON, V. Stream control transmission protocol. RFC 2960, October 2000.

[17] VINOD YEGNESWARAN AND PAUL BARFORD AND SOMESH JHA. Global Intrusion Detection in the DOMINO Overlay System. In *Proceedings of ISOC 2004* (February 2004). `http://www.cs.uwisc.edu/~barford/isoc04.ps`.

[18] WESSELS, D., AND FOMENKOV, M. Wow, that's a lot of packets. In *Proceedings of Active and Passive Measurement Workshop* (April 2003). `http://moat.nlanr.net/PAM2003/PAM2003papers/3836.pdf`.

[19] ZHANG, Y., BRESLAU, L., PAXSON, V., AND SHENKER, S. The characteristics and origins of internet flow rates. In *ACM Sigcomm* (August 2002).

[20] ZOU, C., DUFFIELD, N., TOWSLEY, D., AND GONG, W. Adaptive defense against various network attacks. SRUTI 2005 Workshop (Steps to Reducing Unwanted Traffic on the Internet), Cambridge, MA, July, 2005.