An Extension to Packet Filtering of Programmable Networks

Marcus Schöller, Thomas Gamer, Roland Bless, and Martina Zitterbart

Institut für Telematik
Universität Karlsruhe (TH), Germany

Keywords: Programmable Networks, NodeOS, Packet Selection

Abstract. Several projects proposed to use active or programmable networks to implement attack detection systems for detecting distributed denial of service attacks or worm propagation. In order to distinguish legal traffic from the attack traffic bypassing packets need to be inspected deeply which is resource consuming. Such an inspection can be realized either with additional and expensive special hardware or in software. But due to resource limitations inspection of all bypassing packets in software is not feasible if the packet rate is high. Therefore we propose to add packet selection mechanisms to the NodeOS reference architecture for programmable networks. A packet selector reduces the rate of packets which are inspected. In this paper we detail on various packet selectors and evaluate their suitability for an attack detection system. The results of our implementation show significant advantages by using packet sampling methods compared to packet filtering.

1 Introduction

Distributed denial of service (DDoS) attacks are still a major threat to the Internet today. This is a long known problem to network researchers [8,9] and has attracted public attention since the attacks against Yahoo, CNN, eBay, and many more in the last years. In a major threatening type of DDoS attack the attacker does not exploit a weakness of the victims operating system or application but aims to overload resources like link capacity or memory by flooding the system with more traffic than it can process. The attack traffic is generated by several slave systems which the attacker has compromised before. The attacker has only to coordinate all these slave systems to start the attack nearly at the same time against a single victim. Since the slave systems are scattered over the Internet the attack flows are hard to identify nearby a slave system, because a single attack flow consumes a relative small portion of the overall bandwidth and is therefore indistinguishable from a regular communication flow. On their way towards the victim system the attack flows are aggregated. At this point the attack might be detectable if traffic analysis can be applied. Due to the high bandwidth of backbone links a deep packet inspection of all packets is infeasible even with today's standard router hardware.

Another threat to the Internet today are worms [16,12]. A worm is a piece of software which automatically exploits security holes in operating systems or applications to infiltrate a system. After a successful break-in the worm starts to propagate itself to as many other systems as possible. One side effect of this propagation is the increasing bandwidth consumption since more and more worm instances try to propagate themselves to other systems. In the extreme this can lead to a denial of service attack if the traffic caused by worm propagation overloads the link capacity. Secondly, worms can easily be used to create slave systems for a subsequent DDoS attack. Due to their feature to spread themselves automatically to other systems a large number of slave systems can be aggregated in a relatively short period of time. Today's countermeasures to worms are signature based detection systems scanning for well-known worms. These systems are typically located at the edge of the Internet preventing the worm propagation to a specific network. An earlier detection of such a worm propagation would be possible if the detection system is located in the backbone network. But again a deep packet inspection at backbone rate is needed which is impossible with standard router hardware.

1.1 An attack detection system

One way to realize traffic analysis with deep packet inspection is to enhance the router with special purpose hardware. Another way to implement traffic analysis functionality with deep packet inspection is to use a programmable network like FlexiNet. Such a system allows to create instances of an attack detection system dynamically within the network and the detection system itself can be multi-level.

We designed such an attack detection system to detect DDoS or worm attacks by monitoring the bypassing traffic and retrieving statistical data of predefined aggregates. For each of these aggregates the system computes the average packet count and derives a threshold based on the packet count average and deviation. In normal mode, meaning that no particular attack is currently suspected, only this simple traffic analysis module is running. An excess of the aggregate's threshold indicates an ongoing attack within this aggregate. In this case specialized modules are loaded to analyze the suspicious traffic. These modules can apply various anomaly or signature based tests on the packet stream to verify the attack hypothesis.

As mentioned before the traffic analysis module can not deeply inspect all packets flowing through the system in a backbone network due to resource limitations. Therefore we use a packet selection mechanism [17] to reduce the number of packets requiring deep packet inspection. Based on the statistical data of such samples the same procedures can be applied to find hints on an ongoing attack if the estimation error in each sampling interval remains small. In this paper we will detail on packet selectors in section 2, present an evaluation considering their suitability for an attack detection system and compare the suitable packet selectors with regard to their estimation accuracy.

1.2 Extending the NodeOS specification

The FlexiNet platform was designed according to the NodeOS specification [1] that is, the standard architecture for programmable networks. NodeOS specifies that packet filters for the incoming channel *inChan* must conform to the IPFIX flow definition [14].

This definition allows filtering of packets dependent on the packet's content but it prohibits the implementation of many packet selectors for the incoming channel. The only way to implement a packet selector according to the NodeOS specification is to forward all packets to the execution environment, apply the packet selector there and further process only selected packets. This introduces a high overhead to the systems. This overhead can be reduced if the NodeOS specification gets extended to allow applying packet selectors for the incoming channel.

Therefore we propose to extend the NodeOS specification to allow packet filtering according either to the IPFIX flow definition or to the PSAMP definition, packet sampling according to the PSAMP definition and every combination of these two definitions. The goal of the IETF working group PSAMP is to propose various packet selectors for the Internet especially with the background of traffic measurement.

In contrast to other approaches, that are described in section 1.3 and use programmable networks to build an attack detection system, we base our system on a packet selector in the incoming channel. Besides building an attack detection system the enhancement of the incoming channel enables the creation of services like trajectory sampling, traffic accounting and measurement with reduced overhead.

In section 3 we present implementation details and an evaluation of the most promising selectors for the FlexiNet platform, which we use to detect DDoS attacks and worm propagation in high-speed networks. The results show clearly that the usage of packet selection in the incoming channels reduces significantly the overhead on the system. Therefrom we conclude that an extension of the NodeOS architecture of programmable networks to include the presented sampling method is useful and necessary.

1.3 Related Work

There are some existing approaches which design an attack detection system but they are not using packet selection. In [15] deep packet inspection of all observed packets in a backbone network is achieved by programmable network nodes built of host and network processors. The network processors in these network nodes are able to process packets at line rate so packet selection is not required but this is an expensive approach due to extra special purpose hardware. The approach in [10] does not need deep packet inspection but uses packets which are dropped on a router due to congestion to identify a suspicious aggregate of packets having a certain property and rate limit this high bandwidth aggregate. Furthermore, a pushback mechanism is proposed in which a router can ask upstream routers to control an identified aggregate. This pushback mechanism is achieved by programmable networking.

Packet selection is used in various areas to infer knowledge about an observed packet stream without inspecting all packets. Two examples are the charging from sampled network usage [4] that estimates the user's network usage on the basis of a sampled subset of packets, and the trajectory sampling for direct traffic observation [6] that uses packet filtering to determine the path through a network of a subset of packets.

A performance study of some packet selectors is presented in [3]. The paper compares packet-triggered with time-triggered methods and analyzes the differences between systematic and random selectors. Filtering schemes and probabilistic selectors are not taken into account. The fact that the packet rate of an observed packet stream is not constant over time is addressed in [2]. This paper proposes the usage of an adaptive sampling probability to restrict the sampling error to a predefined tolerance level. This paper does not try to find the optimal choice of a packet selector but addresses a useful optimization in case an optimal packet selector has been chosen.

2 Packet selectors

The IETF PSAMP working group defined two types of packet selectors: filtering and sampling [17], [5]. *Filtering* is used if only a particular subset of packets is of interest. Filtering schemes are always deterministic and are based on packet content or router state. In contrast to filtering, *sampling* is used to infer knowledge about an observed packet stream without inspecting all packets. Therefore only a representative subset of packets is selected which enables an estimation of properties of the unsampled traffic. Sampling methods are either nondeterministic or do not depend on packet content or router state. The sampling methods are further grouped into two categories: random sampling and systematic sampling. First a very brief summary of the filtering schemes and sampling methods is given. A rationale which of the presented methods are suitable for an attack detection system is presented in section 2.3. This subset of methods is examined in section 2.4 in regard to estimation accuracy.

2.1 Filtering schemes

Currently the following three filtering schemes are defined by the IETF PSAMP working group [17]:

– *Field match filtering*—This filtering scheme is based on the IPFIX flow definition. If a specific field of an IP packet matches a predefined value the packet is selected.
– *Hash based selection*—The content of the IP packet or a portion of it is mapped to a hash range using a hash function. A subset of this hash range is defined to be the selection range. A packet is selected if the hash of the current packet is mapped into this selection range.
– *Router state filtering*—A packet is selected if one or more specific states of the router match predefined values. Example states of the router are: ingress interface id, egress interface id, or no route for packet found.

2.2 Sampling methods

Within the sampling methods three indeterministic sampling methods are defined and two methods which are deterministic but independent of packet content and router state.

– *n-out-of-N sampling*—For this method n different numbers must be randomly generated in the range of 1 to N. All packets with a packet position equal to one of the n numbers are selected. This procedure has to be repeated for every interval of N packets.
– *Uniform probabilistic sampling*—Each packet is selected with a fixed uniform probability $1/p$.

- *Non-uniform probabilistic sampling*—This method allows to weight sampling probabilities. Different fixed probabilities can be assigned to different aggregates in order to increase the probability of selecting rare packets.
- *Systematic time based sampling*—A sampling interval is defined consisting of a selection interval and a non-selection interval. A start trigger defines the beginning of a selection interval. All packets arriving after this trigger are selected until the stop trigger fires. No packets are selected thereafter until the new sampling interval starts. After this non-selection interval a new start trigger restarts the method. The unit of the intervals is time based.
- *Systematic count based sampling*—Like systematic time based sampling a selection interval and a non-selection interval are defined. The unit of the intervals is count based. This means that n consecutive packets are selected and the next m packets are not.

2.3 Determining suitable packet selectors for an attack detection system

It is obvious that the presented *filtering* schemes are not suitable for an attack detection system. Any attacker who knows the filtering rules can adapt his attack in a way that his attack packets are not selected by the system. This makes bypassing of the detection system easy.

Non-uniform probabilistic sampling was not taken into account because it needs a deep packet inspection since the selection probability depends on packet content. The systematic time based sampling was not considered either because the estimation accuracy varies on the number of packets during a sampling interval. Additionally the estimation accuracy drops dramatically if the number of packets during the selection interval falls below a threshold and no guarantees about the estimation accuracy can be made.

To implement the n-out-of-N sampling method a list of n unique random numbers must be generated. Therefore, a random number generator must be implemented, memory to save these n numbers must be allocated, and an algorithm to detect duplicate random numbers is needed as well as a sorting algorithm. In contrast, uniform probabilistic sampling requires only a random number generator and memory to save the selection probability. Last, systematic count based sampling requires least resources of all methods. These are memory for the start trigger, the stop trigger, and the packet counter. A problem of uniform probabilistic sampling can be the different number of selected packets in consecutive intervals. This problem vanishes if enough packets during the selection interval are selected. A problem of the systematic count based sampling method is its deterministic approach. If the sampled traffic contains an inherent periodic pattern a detection might fail if the pattern always falls into the non-selection interval.

In summary, we selected the following three sampling methods as interesting candidates for suitable packet selectors and investigated their estimation accuracy: n-out-of-N sampling, uniform probabilistic sampling and systematic count based sampling.

2.4 Estimation accuracy

We compared the suitable packet selectors described in section 2.3 with regard to estimation accuracy. The examination was carried out with an empirically determined sampling probability of 30% which produced acceptable deviations with the used network traces and interval lengths. One packet selector was additionally investigated with sampling probabilities of 20% and 40% to collect some comparative values. The examination used the following different configurations and parameter sets:

i. 30-out-of-100 sampling
ii. 300-out-of-1000 sampling
iii. Uniform probabilistic sampling with a sampling probability of 20%
iv. Uniform probabilistic sampling with a sampling probability of 30%
v. Uniform probabilistic sampling with a sampling probability of 40%
vi. Systematic count based sampling with a selection interval of 3 packets and a non-selection interval of 7 packets
vii. Systematic count based sampling with a selection interval of 30 packets and a non-selection interval of 70 packets

We applied some network traces which originated from the NLANR passive measurement and analysis project [11] to these packet selectors. Therefore, the observed packet stream was divided into intervals with a fixed length and the observed number of packets per interval was examined. The suitable packet selectors were used to infer knowledge about different aggregates like TCP packets, UDP packets etc. without inspecting all packets. The used network traces had a packet rate of about 20 000 packets per second and a duration of about 90 seconds. To make the examined packet selectors comparable the average $\overline{X}$ of the number of packets per interval of one network trace over all observed intervals was calculated for every aggregate. Afterwards the deviation between the original trace, that is when considering all packets, and the sampling run was computed.

$$\text{deviation} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(Y_i - X_i)^2} \tag{1}$$

Y_i represents the estimated number of packets in interval i of the sampling run, X_i counts the number of packets in interval i of the original trace and n counts the number of observed intervals. To get unbiased results 10 sampling runs were carried out per examined packet selector and the average of these 10 runs was calculated. At the end the resulting value was correlated with the calculated average $\overline{X}$ of the original trace

$$\text{deviation}_{rel} = \frac{1}{\overline{X}}\left(\frac{1}{10}\sum_{i=1}^{10}\text{deviation}_i\right) \tag{2}$$

to derive a relative deviation. These relative deviations were used in the following examination to compare the different suitable packet selectors.

Table 1 lists the relative deviations of all examined packet selectors for chosen aggregates and interval lengths. In the first examination an interval length of 5 seconds which corresponds to about 100 000 packets per interval was used. In this scenario we were able to show that with a sampling probability of 30% low bandwidth aggregates like the ICMP aggregate (ICMP 2), which are the worst case in sampling scenarios, only

Table 1. Relative deviations of all examined packet selectors from original traces for chosen aggregates and interval lengths

	Average	30-out-of-100	300-out-of-1000	Uniform 20%	Uniform 30%	Uniform 40%	Systematic 3/7	Systematic 30/70
Interval length: 0.5 seconds								
ICMP 1	87.9	16.25%	16.8%	21.81%	16.47%	13.5%	16.85%	16.11%
UDP 1	1041.95	4.7%	4.47%	6.15%	4.77%	3.8%	4.6%	4.47%
TCP 1	9 343.11	1.03%	0.61%	2.11%	1.54%	1.27%	0.51%	0.7%
Interval length: 5 seconds								
ICMP 2	890.82	5.29%	5.11%	6.84%	5.04%	4.39%	4.09%	4.0%
UDP 2	10 451.29	1.68%	1.53%	2.15%	1.52%	1.27%	1.58%	1.62%
TCP 2	93 423.88	0.9%	0.2%	0.69%	0.49%	0.42%	0.19%	0.21%

have a relative deviation of about 5% from the original trace's values. High bandwidth aggregates like TCP packets (TCP 2) have an even lower relative deviation of under 1% which is an excellent estimation accuracy. In case of a lower interval length of 0.5 seconds, which corresponds to about 10 000 packets per interval, the relative deviation of low bandwidth aggregates degrades to a value of about 16% (ICMP 1) which we found to be acceptable. In this case the relative deviation of the TCP aggregate is about 1% (TCP 1). Using a packet selector with a sampling probability of just 20% (column iii.) the relative deviation degrades in case of the ICMP aggregate to a value over 21% which we did not find acceptable for our attack detection system. With an interval length of 5 seconds the relative deviation of this packet selector would be acceptable, too.

The values in table 1 and the previous results showed that the estimation accuracy can be improved by enlarging the interval length if the packet rate remains constant. But with an attack detection system in mind this surely is not a feasible solution since such a system has to choose the interval length according to detection needs instead of sampling accuracy needs.

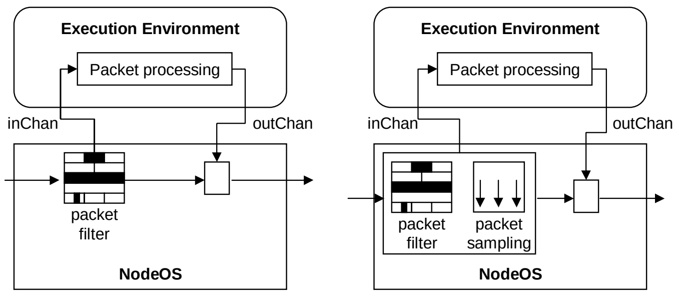
In our examination we could also show that a packet selector with 40% sampling probability (column v.) which has in all aggregates a lower relative deviation than the same selector with a sampling probability of 30% (column iv.) does not improve the estimation accuracy significantly. From these results we could conclude that a higher sampling probability does not justify the higher overhead which arises through the fact that more packets have to be inspected by the attack detection system.

Table 1 also shows that for high bandwidth aggregates 300-out-of-1000 (column ii.) sampling performs slightly better than 30-out-of-100 sampling (column i.) and uniform sampling with the same sampling probability (column iv.). Systematic Count Based Sampling has a similar estimation accuracy than 300-out-of-1000 sampling.

Because all suitable packet selectors have similar estimation accuracies we made our decision on the most suitable packet selector for an attack detection system depending on the required resources like memory or processor time consumption of the different sampling methods. These decision criteria resulted in the usage of systematic count based sampling for packet selection in an attack detection system since this selector needs less resources than n-out-of-N sampling and uniform probabilistic sampling as we already analyzed in section 2.3. Because for systematic count based sampling large selection intervals increase the probability of biased results we always choose the least possible selection interval. In case of a sampling probability of 30% this results in a selection interval of 3 packets and a non-selection interval of 7 packets.

3 Implementation and Evaluation

To implement an attack detection system we used the programmable networking platform FlexiNet [7]. This platform is designed according to the NodeOS specification (see fig. 1a). A service module can install iptables filter [13] rules according to the IPFIX flow definition which select all matching packets in the incoming channel *inChan* and forward them to the FlexiNet execution environment through a netfilter callback function. Having these requirements we implemented in a first approach a NodeOS conforming FlexiNet service module which used systematic count based sampling with a selection interval of 3 packets and a non-selection interval of 7 packets to select packets from an observed packet stream. Then the attack detection system processed the selected packets and at the end of processing every packet was reinjected into normal packet processing through netfilter. The problem with this approach according to the NodeOS specification is that despite of a packet selector all packets of the observed packet stream have to pass through the FlexiNet execution environment since filter rules based on the IPFIX flow definition do not enable packet selection within iptables.



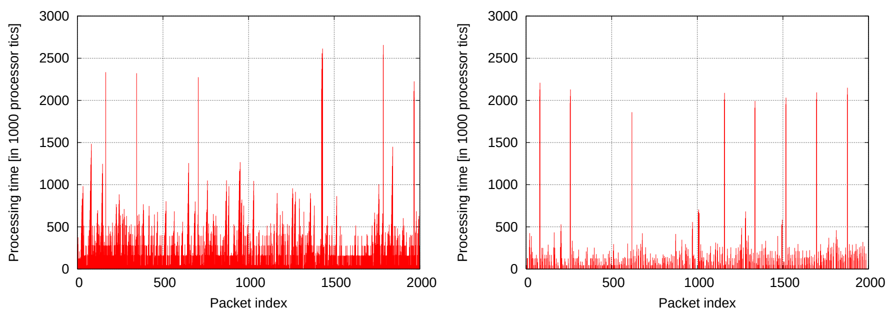
(a) Packet filter for inChan (b) Packet filter and packet selector for inChan

Fig. 1. Proposed extension to the NodeOS reference architecture

A second approach (see fig. 1b) takes the aforementioned problem into account and changes the standard architecture for programmable networks in such a way that packet selection is already possible in the incoming channel *inChan*, that is, before the packets are forwarded to the FlexiNet execution environment. Therefore, the iptables target implementing systematic count based sampling was enabled to apply packet selection instead of forwarding all packets to the FlexiNet execution environment. A copy of every selected packet is queued for later processing while the packet is forwarded normally preserving the overall packet ordering. Packets which are not selected no longer have to pass through the FlexiNet execution environment which results in a significant performance improvement.

3.1 Evaluation

With the implementation of the two packet selection approaches described in the former section we measured the time that was required to process a single packet of the observed packet stream. The *processing time* starts with the check if the observed packet matches the iptables filter rules of the service module and ends with the drop of the copy forwarded to the execution environment. If the packet was not forwarded to the FlexiNet execution environment the processing time ends after the iptables check. The processing time was measured in processor tics by reading out a CPU register through an available C-function. The evaluation was executed on a 2.4 GHz machine so 1 000 processor tics are up to about $0.42\,\mu s$ and 1 ms is up to about 240 000 processor tics, respectively.



(a) Packet selection by ee (b) P[illegible] selection by iptables

Fig. 2. Comparison between processing time of an attack detection system with sampling according to the standard architecture (a) and to an extended architecture (b).

Figure 2(a) shows the processing time needed in case of systematic count based sampling as part of a FlexiNet service module according to the NodeOS reference architecture. In this case packets which are not selected by the sampling method nevertheless have to pass through the FlexiNet execution environment. Figure 2(b) shows the processing time needed in the second approach which implements systematic count based sampling as part of the *inChan* processing. Because of this change of the standard architecture only packets which are selected by the packet selector have to be forwarded to the FlexiNet execution environment. Packets which are not selected by the packet selector can immediately be reinjected into the normal IP packet processing. This behavior results in a significant lower processing time for packets which are not selected by the packet selector in comparison to the processing times in the first approach. This causes the visible gaps between the selection periods in figure 2(b). Selected packets still need a processing time similar to that of the first approach.

The comparison between the minimal processing times of packets which are not selected in the standard architecture of about 250 000 processor tics and in the extended architecture of about 750 processor tics clearly shows that packet selection is a feasible possibility to apply an attack detection system also in backbone networks. This holds since the estimation accuracy of sampling methods is good enough and an extended architecture can save significant processing time for packets which are not selected.

The so far used sampling probability of 30% is still quite large for backbone networks but was chosen due to the pretty low packet rate and duration of the analyzed network trace. In case of the much higher packet rates in backbone networks it is obvious that the sampling probability can be decreased without getting worse estimation accuracy if the interval length remains constant. If we observe for example a packet stream with a packet rate of 500 000 packets per second and use an interval length of 0.5 second we could get approximately the same deviations with a sampling probability of 0.6% than in section 2.4 with a sampling probability of 30% and an interval length of 0.5 seconds.

4 Summary

In this paper we presented various packet selection methods proposed by the IETF PSAMP working group and reasoned which of these are suitable to build an attack detection system in high speed networks. Three sampling methods were considered as suitable and were compared with respect to their estimation accuracy. Since the estimation accuracy of all three methods was similar we preferred the sampling method with fewest resource requirements like CPU and memory.

Further we argued that the current NodeOS specification lacks the possibility to implement packet selection in the incoming channel domain. This introduces unnecessary overhead to the system. In our opinion the NodeOS specification should thus be extended to include sampling methods in the incoming channel domain. This extension allows to build active and programmable networks for attack detection systems as well as for traffic measurement or other applications of packet selection. We further described such an extended programmable network and showed implementation results proving the advantages of our proposal.

References

1. Active Networking NodeOS Working Group. NodeOS Interface Specification, Jan 2002. Available from www.lancs.ac.uk/postgrad/bourakis/papers/an_node_.pdf.

2. B.-Y. Choi, J. Park, and Z.-L. Zhang. Adaptive random sampling for load change detection. In *SIGMETRICS '02: Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 272–273, New York, NY, USA, 2002. ACM Press.
3. K. C. Claffy, G. C. Polyzos, and H.-W. Braun. Application of sampling methodologies to network traffic characterization. *SIGCOMM Comput. Commun. Rev.*, 23(4):194–203, 1993.
4. N. Duffield, C. Lund, and M. Thorup. Charging from sampled network usage. In *IMW '01: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 245–256, New York, NY, USA, 2001. ACM Press.
5. N. G. Duffield. A framework for packet selection and reporting. Internet Draft, draft-ietf-psamp-framework-10.txt, Work in Progress, Internet Engineering Task Force, January 2005.
6. N. G. Duffield and M. Grossglauser. Trajectory sampling for direct traffic observation. In *Proceedings of SIGCOMM*, pages 271–282, 2000.
7. T. Fuhrmann, T. Harbaum, M. Schöller, and M. Zitterbart. AMnet 3.0 source code distribution. Available from http://www.flexinet.de.
8. L. Garber. Denial-of-service attacks rip the internet. *Computer*, 33(4):12–17, 2000.
9. A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks-extended. Technical Report ISI-TR-2003-569b, USC/Information Sciences Institute, June 2003. (Original TR, February 2003, updated June 2003).
10. R. Mahajan, S. Bellovin, S. Floyd, J. Vern, and P. Scott. Controlling high bandwidth aggregates in the network, 2001.
11. N. Measurement and N. A. Group. http://pma.nlanr.net.
12. D. Moore, C. Shannon, and K. C. Claffy. Code-red: a case study on the spread and victims of an internet worm. In *Internet Measurement Workshop*, pages 273–284, 2002.
13. T. netfilter/iptables project. http://www.iptables.org.
14. J. Quittek, S. Bryant, B. Claise, and J.Meyer. Information model for ip flow information export. Internet Draft, draft-ietf-ipfix-info-07.txt, Work in Progress, Internet Engineering Task Force, May 2005.
15. L. Ruf, A. Wagner, K. Farkas, and B. Plattner. A Detection And Filter System for Use Against Large-Scale DDoS Attacks In the Internet-Backbone. In *Proc. of 6th Annual Int. Working Conf. on Active Networking (IWAN), Lawrence Kansas, USA*, Lecture Notes in Computer Science. Springer Verlag, Heidelberg, Oct. 2004.
16. C. Shannon and D. Moore. The spread of the witty worm. *IEEE Security and Privacy*, 2(4):46–50, 2004.
17. T. Zseby, M. Molina, F. Raspall, and N. G. Duffield. Sampling and filtering techniques for ip packet selection. Internet Draft, draft-ietf-psamp-sample-tech-07.txt, Work in Progress, Internet Engineering Task Force, July 2005.