COPS: Quality of Service vs. Any Service at All

Randy Katz[1], George Porter, Scott Shenker, Ion Stoica, Mel Tsai

University of California, Berkeley
{randy, gporter, istoica, shenker, mtsai}@cs.berkeley.edu

Abstract. Today's networks are awash in illegitimate traffic: port scans, propagating worms, and illegal peer-to-peer transfers of materials [1]. This "noise" has created such a crescendo that legitimate traffic is starved for network resources. Essential network services, like DNS and remote file systems, are rendered unavailable. The challenge is no longer "quality of service" but rather "any service at all." Techniques must be developed to identify and segregate traffic into good, bad, and suspicious classes. Quality of Service should now protect the good, block the bad, and slow the ugly when the network is under stress of high resource utilization. We discuss the research challenges and outline a possible architectural approach: COPS (Checking, Observing, and Protecting Services). It is founded on *Inspection-and-Action Boxes* (iBoxes) and *packet annotations*. The former are middlebox network elements able to inspect packets deeply while performing filtering, shaping, and labelling actions upon them. The latter is a new layer between routing and transport that tags packets for control purposes while also providing an in-band control plane for managing iBoxes across a network.

1 Introduction

Networks have become critical for the proper functioning of modern enterprises. An enterprise must be able to depend on its network—the part it operates, as well as the rest to which it interfaces—to provide reliable end-to-end connectivity between consumers, suppliers, and for wide-area applications. Dependability encompasses *reliability* (i.e., a path can always be established between end-points even in the face of link and router failures) and *trustability* (i.e., reachability can be achieved even in the face of incorrect or malicious behaviour within the network). One down hour of access to Amazon.com results in an estimated loss of revenue of $600,000 [2]. Being able to depend on the network when you need it is a critical requirement for modern network-based applications.

Traditional quality of service methods focus on improving network performance by managing latency and bandwidth. Little effort has been directed towards improving the dependability of networked systems. Yet weaknesses in dependability seriously impacts performance. For example, denial of service attacks generate so much traffic that

1. *Author's address:* 637 Soda Hall, CS Division, EECS Department, University of California, Berkeley, CA 94720-1776 USA. *Tel:* +1 (510) 642.8778. *Fax:* +1 (510) 642.5775. *E-mail:* randy@cs.Berkeley.edu. *WWW:* http://www.cs.Berkeley.edu/~randy.

the normal functioning of the network collapses, yielding link resets and the melt-down of routing protocols in the wide-area. In the local-area, critical applications services like name servers and remote file systems are rendered inaccessible. Poor reliability and poor trust translate into poor performance and poor service.

The critical performance challenge is no longer service differentiation; rather, it is to enhance the dependability of networked applications while protecting the network-based services upon which they rely. Any method likely to succeed must be able to detect unusual network behaviours—like unanticipated traffic surges—and perform actions to correct or recover from these. Our approach is founded on two components. First, we introduce *Inspection-and-Action Boxes* (iBoxes): network elements able to observe traffic and act upon it by filtering and shaping traffic. iBoxes are built from *programmable network elements* (PNEs)—Layer 2 and 3 devices with enhanced packet processing via flexible packet classification, transformation, and action while operating at network line speeds. Second, we introduce a new *Annotation Layer* between routing and transport to enable information sharing among iBoxes. Together these provide the essential foundation for (enterprise) network-wide Observe-Analyze-Action.

The rest of the paper is organized as follows. First we present some background on the causes of network failures and prior work on extensible networking. In Section 3, we introduce COPS, our conceptual approach for checking-observing-protecting network services. Section 4 describes our Inspection-and-Action Boxes, which provide the observation-and-action points within the network to implement COPS. We introduce the annotation layer in Section 5, and illustrate its use in Section 6 for a network management and protection application. Our summary and conclusions are in Section 7.

2 Background and Related Work

2.1 Network Protection and Quality of Service

A key challenge for the administrators of the Berkeley Campus Network, a diverse edge network with up to 40,000 active ports on an average day, is dealing with unanticipated traffic surges that render the network unmanageable [3]. These surges can be due to a denial of service attack, the outbreak of the latest Internet worm, or a new file sharing protocol recently discovered by the students. While the source of the surge is often difficult to determine initially, the way it leads to service failure is remarkably similar: the in-band control channel is starved, making it difficult to manage and recover the network exactly when you most need to do so.

Another example is offered by the administrators of our departmental network. In mid-December 2004, largely after the students had left for Christmas break, traffic surges rendered DNS and remote file systems unusable [4]. While the administrators suspected a denial-of-service attack against DNS at the time, another possible source was a poorly implemented and configured spam appliance that generates DNS queries for every email message it examines. Even to this day, and after extensive examination of system logs, the administrators have yet to identify the true source of the surge.

Traditional security tools like intrusion detection systems, firewalls, and virus detection software offer only limited protection in the situations suggested by the above. The signature of events that generate the surge are unlikely to be found in the existing fault databases of these systems. The source of the surge is as likely to be inside the network as outside, and so boundary methods like firewalls are insufficient in any event. Furthermore, the root cause of the surge may be difficult to identify, and need not have anything to do with overtly malicious attacks against the network.

Traditional QoS mechanisms, like DiffServ and IntServ, allocate network resources like bandwidth to statically identified traffic classes [5]. An example is rate-limiting UDP traffic to reserve sufficient bandwidth for TCP traffic. In our view, since the threats are evolving too quickly for static traffic characterization and blocking, protecting networks from unanticipated surges isn't about strict reservations and resource allocation policies. Rather, we need survival policies when the network is under stress.

2.2 Active Networking and Commercial Network Appliances

There is a comprehensive research literature on *active networking*, first proposed by David Tennenhouse [6]. Active networks allow third parties to inject their own functionality into distributed network nodes. The concept remains highly controversial, because of the numerous performance and access control challenges it raises. From our perspective, we are not concerned with running arbitrary application code inside the network. Others have focused on applying programmable networks for network management [7]. Our focus is on protective mechanisms to defend critical network services.

As evidenced by the emergence of *network appliances* for network functions like firewalls, intrusion detection systems, network address translation, traffic blocking and shaping, spam filtering, storage virtualization, and server load balancing, deep packet analysis and processing at the network layer is a commercial reality. Unfortunately, existing appliances focus on point functionality, and they do not offer a network-wide platform upon which to develop a comprehensive architecture for network protection.

We are interested in network management and control mechanisms that become operational when the network detects that it is under stress. We believe that we can build such a capability on top of commercially available PNEs. We concentrate on enterprise-area network management, where the services to be protected are simpler to identify and the policies for their protection can be easily specified.

2.3 Extensible Routing

Extensible routing provides one method for adding capabilities to the network. *Click* is a modular software router developed at MIT [8] and extended by the *XORP* open source effort [9]. A Click router is a collection of modules called *elements*. These control a router's behaviour, including operations like packet modification, packet queuing, packet dropping and packet scheduling. A new router is implemented by gluing together elements using a simple configuration language. Click/XORP provides a powerful framework for developing new and extended software-based implementations of network protocols, but it provides no particular platform support for network-wide protection. It is an excellent prototyping environment for packet processing experimentation, but it lacks an implementation path to performance commensurate with the gigabit and higher line speeds of modern local-area networks.

3 COPS Paradigm

The Internet's protocols were designed to tolerate point failures, such as a broken router or link. Failures induced by syntactically well-formed traffic that is otherwise badly behaved were never formally considered. Protocols are vulnerable to easy exploitation, through Denial of Service attacks that overwhelm a service, inducing traffic loads that starve the control plane of legitimate service requests. While the phenomenology of failure is complex, the recent Berkeley experience discussed in Section 2 suggests that it is the effect of unanticipated traffic surges that render enterprise networks unusable.

What is needed is a comprehensive architecture for network protection. Under times of high utilization, our approach prioritizes network resources for good traffic, blocks known bad traffic, and slows suspicious traffic until it can be further classified. To succeed, it is crucial that the behaviour of the network be checkable and observable. Network protocols are *checkable* if their behaviour can be verified as being well-formed and semantically consistent. This simplifies the job of identifying good traffic. Most existing behaviours have not been designed to be checkable, so we must *observe* them over time to infer their good or bad qualities. Once classified as good, bad, or ugly (i.e., still to be determined), we *protect* network resources by limiting ugly flows while guaranteeing bandwidth to control plane messages. We describe our approach in more detail in the following subsections.

3.1 Checking

Protocols whose behaviour can be characterized by an invariant can be checked by network entities. An invariant is not foolproof, so there is no guarantee of correct protocol behaviour even if it holds. Rather, our goal is to provide significant but imperfect protection against misconfigurations or malicious attacks. The approach uses a network entity to observe a protocol's traffic, perhaps using statistical sampling. It checks the protocol invariant to determine if the protocol is semantically-consistent and well-behaved. It protects well-behaved traffic, by lowering the priority of suspect traffic.

Checkable protocols often require new protocols or significant changes to the endpoints. Due to limited space, we do not present our approach to the design of checkable protocols in detail. However, we have developed checkable versions of BGP called Listen and Whisper [10] and QoS-based Traffic Rate Control [11]. These designs offer significant building blocks that can be used in constructing checkable protocols: observable protocol behaviour, cryptographic techniques, and statistical methods.

3.2 Observing

Achieving protection of network services requires "in-the-network" observation and control points. Such protocol-aware observation points can detect inappropriate traffic, such as a SYN flood or a DNS smurf attack. When traffic is too high, thus threatening critical services, these points invoke protective mechanisms. At a basic level, observation points passively collect data and calculate statistics to distinguish between normal operation and a network under stress. While this is not unlike existing network tools for data collection, we can combine observation with some analysis to enable management actions. Because we can inspect packets, thus achieving a degree of protocol awareness, these are points "inside-the-network" to verify checkable protocol invariants. A checkable protocol flow for which an observation point has detected an invariant violation can either restore the invariant (e.g., "punish" the flow by restoring it to a fair rate) or even drop it before its reaches a vulnerable network service.

Beyond filtering and traffic shaping, these points can perform deeper actions on packets. For example, they can add annotations to support new network controls, like inserting labels in packets entering the enterprise at the Internet edge to distinguish among outside packets and those generated internally. This enables a spectrum of new protective actions beyond traditional firewalls or traffic managers: e.g., when a critical resource is overloaded, they preferentially queue local traffic, thereby deferring resources from external traffic. This will be described in more detail in Section 5.

3.3 Protecting

The conventional approach to avoiding attacks is to identify and then block bad traffic. Such distinctions are difficult. To be effective, almost all bad traffic must be identified. To make the defense acceptably tolerant, almost all good traffic must be passed. This sets an impossibly high bar for classification, as very few mistakes can be tolerated. We adopt a different approach: protect crucial network services (e.g., DNS, a network file system, etc.) for use by crucial and trusted clients (e.g., authenticated end-hosts of the organization versus outside sources). Identifying a set of trusted clients beforehand, and verifying that identity, simplifies the task of characterizing traffic into "known good and important," "known bad," and "unknown."

We block "known to be bad" traffic but also ensure that "known to be good" traffic receives sufficient resources at critical services. Ugly traffic is not blocked, but network resources are allocated so they cannot prevent good traffic from getting their required services. This triage simplifies the problem. The identification of bad traffic can be conservative, since very high accuracy isn't needed to preserve the network's functioning. We can use explicit signals to identify good traffic, based on the annotation layer. This relieves the network from making real-time judgments about the nature of the traffic; instead, decisions about goodness can be based on long-standing and operator-tunable policies. We do not seek perfect protection; instead, we only minimize and mitigate the effect of attacks or other traffic surges that can affect the availability of network services. With our approach, attacks may cripple non-essential services and/or block non-essential users, but the use of crucial services by essential users is protected.

4 Inspection-and-Action Boxes

4.1 Programmable Network Elements

While specific PNEs vary in architecture, there is much they share in common. Figure 1 offers a generic block diagram consisting of input and output ports and buffer memories interconnected by a high speed interconnection fabric, and Classification (CP) and Action (AP) Processors. Packets are presented at an input port, where they are staged into a buffer memory. One of several classification processors examines the packet based on a set of specified pattern matching and sequencing rules. The multiple CPs operate in parallel. They can inspect packet fields beyond headers. The complexity of the

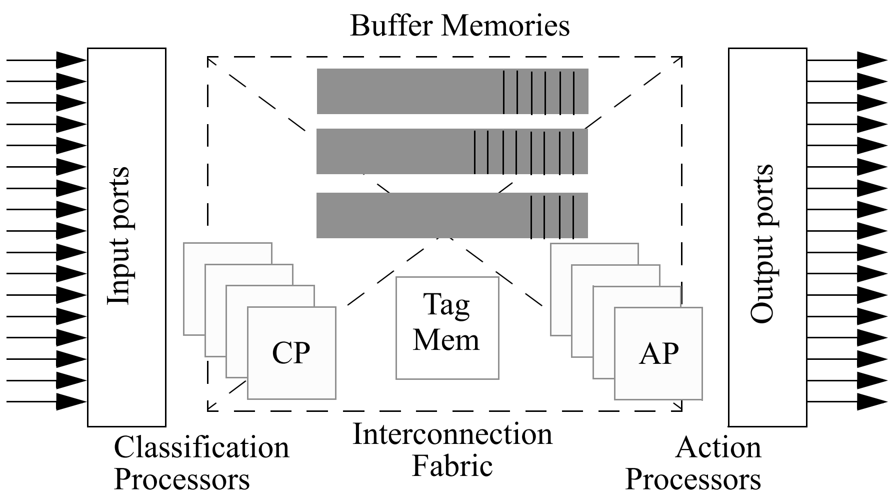


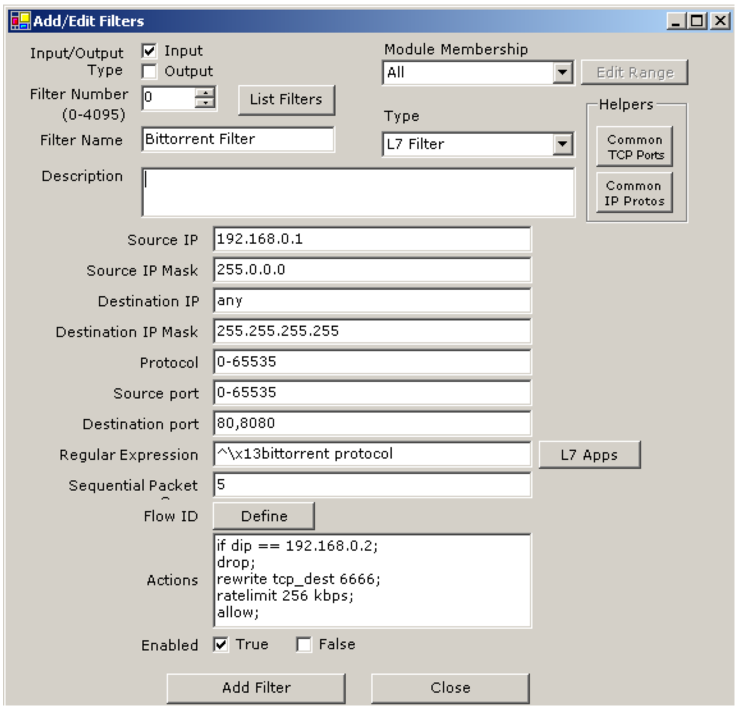
Figure 1 Generic Internal Organization of a Programmable Network Element

classification rules depends on how deep within protocol layers the packet analysis requires. Once a classification decision is made, tagging information is associated with the packet in a Tag Memory (TM). An AP can now process the packet and its tags. It moves the packet to an output queue, modifying specific fields or generating new packets in response to detecting this particular packet, either to be queued for transmission to the sender or to be forwarded on to the receiver. It retains information about the packet and the session or flow to which it belongs. Finally, the packet is queued for output to a particular port, based on the policies and processing done at the APs.

4.2 RouterVM

RouterVM is our mechanism for specifying packet processing [12]. It is constructed from cascaded generalized packet filters, a bundling of *patterns* for packet classification and *actions* for manipulating packets that match these patterns. Figure 2 illustrates the concept with the user interface used to specify GPFs. A GPF can specify packet contents beyond IP headers, allowing the GPF author to express higher level protocol patterns, such as the embedded *bittorrent* file sharing protocol headers in the figure (this is accomplished via a regular expression found in an attached library). An action language allows us to further control packet flow. In the example action list shown in the figure, packets for IP address 192.168.0.2 are dropped, while the remaining packets are directed to port 6666 and rate-limited to 256 kbps.

To test the generality and flexibility of GPFs, and to develop a better understanding of how state needs to managed in such an environment, we have written proof-of-concept specifications for several network observe-analyze-act functions. These include traffic shaping and monitoring, layer 7 traffic detection (e.g., recognizing Kazaa, HTTP, AIM, POP3, etc. traffic flows), quality of services and packet scheduling, network address translation, intrusion detection, protocol conversion (e.g., IPv6-to-IPv4 interworking), content caching, server load balancing, router/server health monitoring, stor-


Add/Edit Filters

Input/Output Type	[✓ Input] Output
Filter Number (0-4095)	0
Filter Name	Bittorrent Filter
Description	
Module Membership	All
Type	L7 Filter
Source IP	192.168.0.1
Source IP Mask	255.0.0.0
Destination IP	any
Destination IP Mask	255.255.255.255
Protocol	0-65535
Source port	0-65535
Destination port	80,8080
Regular Expression	^\x13bittorrent protocol
Sequential Packet	5
Enabled	[✓ True] False

Actions:

```
if dip == 192.168.0.2;
drop;
rewrite tcp_dest 6666;
ratelimit 256 kbps;
allow;
```

Figure 2 Generalized Packet Filters

age networking (including iSCSI management), fibre channel to IP interworking, iSCSI, XML preprocessing, TCP offloading, mobile host management (e.g., 802.11), encryption/decryption for virtual private networks, and network data structures (e.g., multicast, overlays, and distributed hash tables).

The essential building blocks within GPF specifications to implement these functions are illustrated with the following examples:

- *Programmatic decision making* (e.g., "if dest_ip == 127.0.0.0 then drop;"). An essential feature of GPFs is the ability to perform an action based on a general classification of a packet's content.
- *Server load balancing* (e.g., "loadbalance table SLB_Table;"). Tables are GPF's data structure for maintaining state. An element of the loadbalance action is to examine the server load balance (SLB) table to assess currently assigned loads to individual servers as an input to the decision process of allocating the next flow.
- *Packet field rewriting* (e.g., "rewrite dest_ip 192.168.0.1;"). Rewriting actions is the building block for implementing a variety of translation actions such as NAT and support for mobility.
- *Packet duplication* (e.g., "copy;"). Packet duplication is a useful building block for numerous operations, such as updating mirrored storage or to spawn a packet for redirection to another point in the network, like an intrusion detection system.

- *QoS* (e.g., “ratelimit 1 Mbps;”). This building block identifies flows and schedules them in such a fashion as to achieve a target rate limit.
- *Packet logging* (e.g., “log intrusion_log.txt;”). Packets and packet subsets can be saved to log data structures for later analysis.
- *Network address translation* (e.g., “nat dir=forward, table=NAT_table;”). NAT actions are constructed as a combination of table look-up and field re-writing.
- *Server health monitoring* (e.g., “if 192.168.0.5 is alive;”). Network node liveness detection can be used as a building block in triggering actions.

This is not an exhaustive description of GPFs, but rather a taste of the classification and actions from which network functions can be specified. We are continuing to define GPFs and to study the issues in mapping such specifications into target implementations. Our goal is to understand how flexibility of expression influences implementation cost, and how such costs could be mitigated by underlying hardware support.

4.3 iBox Placement

To emphasize how PNEs form a foundation for implementing inspection and action at critical points within edge networks, we call them *iBoxes* when used in this way. Figure 3 illustrates the most likely places for iBoxes within an Enterprise network. The server and user edges are interconnected by the Distribution tier to the Internet edge, providing connectivity to the network outside of the immediate administrative control of the enterprise. Rs are routers, Es are end nodes, and Is represent iBoxes. iBoxes are placed between existing routers and in front of network and user services, like DNS and file servers.

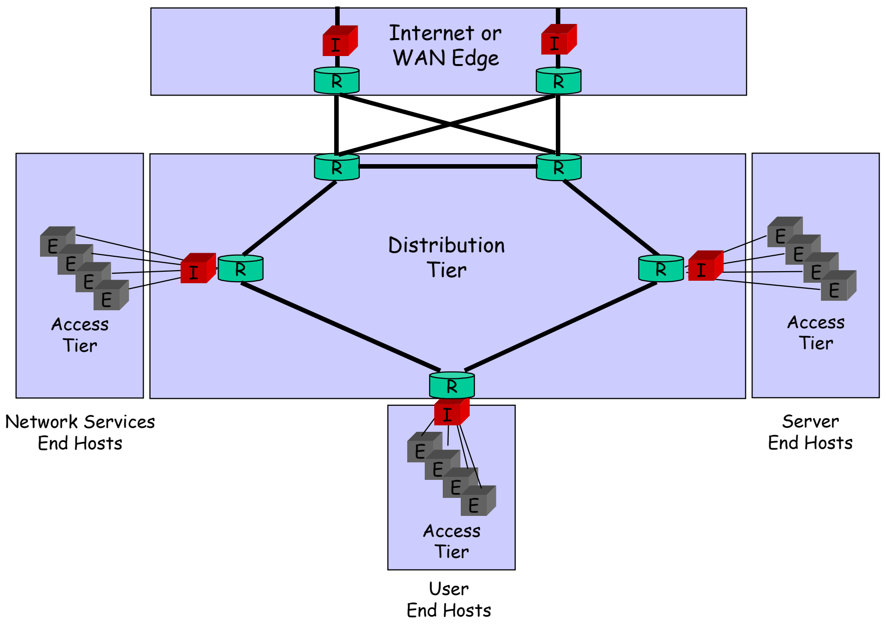


Figure 3 Enterprise Network Placement of iBoxes

There is value in off-loading "observation" from routers, thus allowing existing routers to be extended by cascading them with iBoxes. Because commercial network appliances—upon which iBoxes are ultimately implemented—have been "built" for inspection and analysis, this helps iBoxes avoid denial of service attacks based on traffic diversity/slow path processing known to plague conventional routers. Since iBoxes seek to detect traffic surges, statistical sampling techniques offer a feasible approach for reducing the classification load during periods of high traffic. However, as an edge network technology, iBoxes cannot help an enterprise's flooded Internet connection.

iBoxes perform management actions through coordination and sharing of information. Observations are summarized and statistical abstracts shared among iBoxes within a given network. Whether a distributed localized algorithm for action decision-making is sufficient, or a centralized/hierarchical scheme scales better is under investigation. A mechanism is needed to allow iBoxes to intercommunicate. This is the essential functionality provided by the Annotation Layer, presented next. iBoxes assign a high priority and reserve bandwidth for their own signalling annotations to insure that the network remains manageable even in the face of surging traffic.

5 Annotation Layer

Many network appliances (e.g., NAT boxes, server load balancers), rewrite packet headers, thus violating the Internet's end-to-end principle. Encryption/decryption devices change packet contents, as do compression/decompression boxes. Given a desire to retain network layerization, yet exploit deeper inspection to better analyze packet contents, we introduce a new annotation layer between routing and transport. Figure 4 illustrates our approach. iBoxes insert annotation labels into packets summarizing the results of their analyses. At their simplest, these capture straightforward packet attributes (e.g., whether they were generated within the network or without). With the network under stress, a simple policy passes internally generated packets to local network services, while externally labelled packets are slowed/dropped. More sophisticated examples include the summarization of how packets have been classified into good, bad, and ugly along with statistics that characterize the nature of the flow to which it belongs.

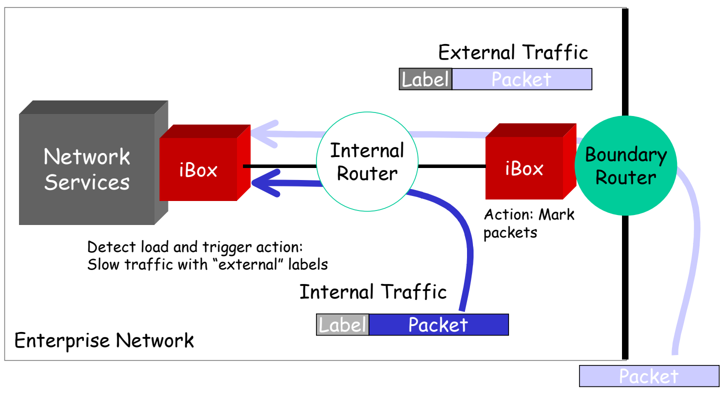


Figure 4 iBox Annotation using Labels

A possible impediment to in-depth packet inspection is the widespread deployment of IP security. One possibility is to make iBoxes part of the trusted infrastructure, with access to end-host keys. iBoxes could then decrypt, inspect, reencrypt, label, and pass packets on. Such an approach may not be feasible in all networks (in addition to the processing overheads). In that case, inspection and subsequent labelling can be performed for those coarse attributes gleaned from the packet flows, such as packet size, packet interarrival times organized by sources and destinations, and frequency of encountering given source and IP address ranges. This is not an exhaustive list, but rather represents the kinds of attributes that can be extracted even from encrypted streams. We plan to demonstrate how to distinguish and protect the good flows, even if it becomes more difficult to identify the bad flows.

6 Network Management and Service Protection

6.1 General Approach

Better network reliability is founded on mechanisms monitoring network performance, and rapidly detecting and reacting to failures. One approach is *active probing,* a technique that periodically sends requests to a server or network element to measure the network-level response time. However, this does not scale due to the overhead of injecting measurement traffic. Passive monitoring is more attractive. It is inherently scalable and it detects network failures more quickly in parts of the network more frequently accessed. These usually represent the more critical network paths. Previous work on passive monitoring was limited by the weak support provided by network hardware [13]. PNEs provide a new platform for deploying passive monitoring within edge networks. Their positioning within edge networks is an advantage; they are easier to control, and when multiple devices cooperate, it is possible to better pinpoint network failures.

We are developing a shared monitoring infrastructure to improve network reliability. This is based on applying statistical learning theory [14] to build models of expected performance, using passive measurements collected over time. Such a performance profile consists of a distribution of network and application-level measurements like roundtrip time, packet loss, throughput, and application requests, organized according to network address prefix. We use this to detect performance degradation by calculating the probability of consecutive network events. A small probability implies a performance change, which may indicate failed or failing network paths. We are also investigating how various monitoring points can share their measurements via the annotation layer, thus accelerating learning while also correlating observations for fault diagnosis.

6.2 A Scenario: DNS Service Degradation and Protection

In this subsection, we describe how the December 2004 loss of service experienced at Berkeley could have been mitigated by the COPS approach, and in so doing, illustrate the concepts we have presented above. First we consider the case of an externally generated DoS attack against the DNS service and then we examine the alternate case of the misconfigured spam appliance.

Figure 5 shows the network topology, with iBoxes placed at the Internet, Access, and Server edges of the network (labelled I_I, I_A, and I_S respectively). Through protocol-

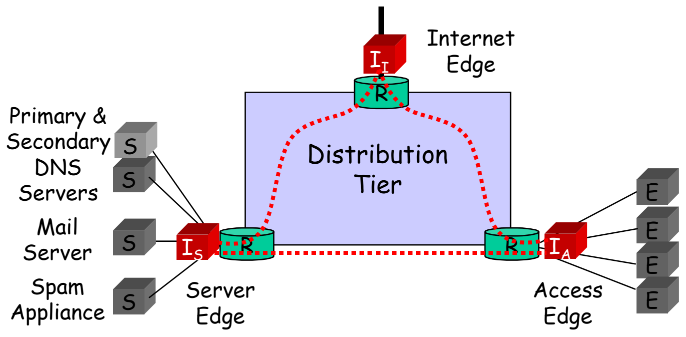


Figure 5 Network Topology for Scenario

aware packet inspection and statistics collection, I_S detects an unusual increase in latency between the arrival of DNS queries and their response. As other services are within normal ranges, it can pinpoint the problem to the DNS server. It exchanges information with I_I via the annotation layer to determine if the number of external DNS queries is high. If so, I_I negotiates with I_S to slow these requests to a level that insures that internally generated requests receive adequate service. Alternatively, I_S can load balance requests to the Primary and Secondary DNS servers, redirecting external requests to the latter while local requests are sent to the former. Such a strategy only works if indeed it is the server rather than the server network segment that is the performance bottleneck.

Another theory for our DNS failure is that our third party spam appliance generated a large number of DNS requests to verify mail domain validity during an email flood. An unusual increase in email traffic positively correlates with increased DNS latencies, thereby affecting the performance of web access and network file system access in a completely unexpected way. Yet even in this difficult case for human diagnosis, the iBoxes can detect such correlations, pinpoint the email surge as the root cause of the problem, and slow email delivery to regain control of the DNS service.

The scenario illustrates some points about network protection. Slowing email to restore DNS availability is a policy decision that must be determined in advance, or presented to the network administrators for their adjudication. Observability also affects the network topology design. Our iBox placement makes it difficult to observe/act on the traffic between the spam appliance and the mail and DNS servers. Had the network been designed so that orthogonal services were placed in different segments visible to iBoxes, the infrastructure could then detect the unusual number of DNS queries originating at the spam appliance. Given sufficient protocol awareness, one possible action is to bypass the spam appliance altogether. A better alternative is to spawn a new DNS server in response to the increased demand for domain name lookups, while redirecting to it some of the traffic load. Our administrators essentially did this by hand: our surge problems went away when they dedicated a new DNS server to the spam appliance.

7 Summary and Conclusions

With the emergence of a broad range of network appliances for diverse packet processing, it is clear that the active networking promise of Processing-in-the-Network is now a reality. While these devices succeed in integrating networking and processing, they still lack a unified framework for specifying and extending their functionality. We have been developing such a framework based on *RouterVM*, to describe packet filtering, redirection, and transmission, with additional mechanisms to enable session extraction and packet execution based on session-level context.

We believe that the challenge of protecting network services when the network is under stress can be met by harnessing these programmable network elements for a pervasive infrastructure for observation and action at the network level. In implementing the Check-Observe-Protect paradigm, PNEs form the foundation of *iBox*-based "inside the network" observation and action points, and the *Annotation Layer* provides the mechanism for inter-iBox coordination. While for today's Internet, we have introduced iBoxes as standalone networks elements, there is no reason to believe that their functionality could not eventually be migrated into future router architectures.

References

[1] R. Pang, V. Yegneswaran, P Barford, V. Paxson, L. Peterson, "Characteristics of Internet Background Radiation," ACM Internet Measurement Conference 2004, Taormina, Sicily, (October 2004).

[2] A. Fox, D. Patterson, "Self-Repairing Computers," Scientific American, (June 2003). URL: http://www.sciam.com/article.cfm?colID=1&articleID=000DAA41-3B4E-1EB7-BDC0809EC588EEDF.

[3] Personal Communications, Berkeley IS&T Staff, (August 2004).

[4] Personal Communications, Berkeley EECS Network Administrators, (February 2005).

[5] RFC 2998, "A Framework for Integrated Services Operation over Diffserv Networks."

[6] D. L. Tennenhouse, D. J. Wetherall, "Towards an Active Network Architecture," *Computer Communications Review*, V. 26, N. 2, (April 1996).

[7] A. Galis, S. Denazis, C. Brou, C. Klein, eds., *Programmable Networks for IP Service Deployment*, Artech House Publishers, London, (2004).

[8] E. Kohler, R. Morris, B. Chen, J. Jannotti, M. F. Kaashoek, "The Click Modular Router," *ACM Transactions on Computer Systems*, vol. 18, no. 4, (November 2000).

[9] http://www.xorp.org.

[10] L. Subramanian, V. Roth, I. Stoica, R. H. Katz, S. Shenker, "Listen and Whisper: Security Mechanisms for BGP," USENIX/ACM Symposium on Networked System Design and Implementation (NSDI'04), San Francisco, CA, (March 2004).

[11] I. Stoica, H. Zhang, S. Shenker, "Self-Verifying CSFQ," Proceedings of INFOCOM'02, New York, (June 2002), pp. 21-30.

[12] M. Tsai, "The Design and Implementation of RouterVM," Ph.D. Dissertation, U. C. Berkeley, (expected August 2005).

[13] M. Stemm, S. Seshan, R. H. Katz, "A Network Measurement Architecture for Adaptive Applications," IEEE Infocomm 2000 Conference, Tel Aviv, Israel, (March 2000).

[14] A. X. Zheng, M. I. Jordan, B. Liblit, A. Aiken, "Statistical debugging of sampled programs," *Advances in Neural Information Processing Systems (NIPS)* 16, 2003.