

Anemone: using end-systems as a rich network management platform

Microsoft Technical Report, MSR-TR-2005-62

Richard Mortier, Rebecca Isaacs, Paul Barham
Microsoft Research, Cambridge, UK

May 20, 2005

Abstract

Enterprise networks contain hundreds, if not thousands, of cooperative end-systems. This paper advocates devoting a small fraction of their idle cycles, free disk space and network bandwidth to create *Anemone*, a rich platform for network management. This contrasts with current approaches that rely on traffic statistics provided by network devices. Anemone infers network-wide traffic patterns by synthesizing end-system flow statistics with dynamic topology information obtained by passive snooping of IP routing protocols. Understanding the effect of the network on individual applications' end-to-end performance requires data only available in the end-systems actually hosting applications. Consequently, we claim that augmenting end-systems with in-band monitoring, creating an overlay of real-time 'traffic sensors' will provide a more complete view of the network, support sophisticated network management queries, and supply the global statistics necessary to automate network control. This paper describes Anemone, discusses potential benefits and challenges, and presents an initial simulation of the platform.

1 Introduction

Many network management tasks require deeper understanding of the state of the network that can be acquired solely from information available in the core. Modern networks are deploying more and more protocols that are difficult to track from the core of the network due to the use of tunnelling and encryption. The effect of the network on an individual application's end-to-end performance (e.g. the delay associated with a VoIP call), requires data that is only available in the end-systems actually hosting the application (e.g. IPSec decryption keys). Consequently, we have designed a network management platform called *Anemone* that combines end-system instrumentation with routing protocol collection to provide a semantically rich view of the network.

Essentially, Anemone treats the end-systems in the network as a set of 'traffic sensors' and combines flow data from these systems with topology data inferred from the routing protocol to provide a rich dataset for

mining by network management applications. Initially, we focus on enterprise networks, exploiting their centralized host administration to get a coherent picture of the traffic entering and leaving the network. Due to the nature of the instrumentation, this dataset not only avoids problems due to protocols that are opaque to devices in the network core, but also exposes both then application-level semantics of traffic and its treatment inside the network.

End-systems are a highly appropriate place to locate support for flow-based network management, having three main benefits over traditional SNMP-based network management tools:

1. As noted above, opaque protocols that use tunnelling and encryption are becoming widely deployed in enterprise IP networks, and furthermore, core routers are often resource starved. As a result, in-core monitoring tools such as NetFlow [4] are becoming less useful due to the increase in traffic that they simply cannot usefully examine. Moreover, even where they can gather useful data, the overheads they impose typically require them to be configured to use sampling techniques rather than continuous monitoring [8].

In contrast, each end-system has all the information (e.g. IPSec keys) required to decrypt and de-encapsulate the traffic it transmits and receives, as well as having CPU, memory and disk resources to spare. The result is that they can collect more detailed and relevant data about traffic, and can store this data for long periods of time enabling useful applications such as capacity planning and anomaly detection.

2. Modern networks often have tens, if not hundreds, of types of network device comprising different hardware/firmware/operating system combinations. These often have subtle differences, and frequently bugs, in their MIBs (Management Information Bases – database tables), as they are not critical to the operation of the device.

In contrast, most enterprise networks typically have only a handful of types of end-system, reducing the severity of this problem. Furthermore, use of the routing protocol to recover topology information is relatively sound since this assumes correctness of only the routing protocol and IP packet forwarding implementations. Since these are critical for correct functioning of any router device, they are likely to be less buggy than non-critical SNMP MIB implementations.

3. Even where all traffic is transparent to network core monitoring systems and such systems may be continuously deployed, it can be difficult to determine application-perceived performance from network measurements. Dynamic port negotiation, the need for temporal correlation between flows, and client/server performance variation all make it difficult to understand the impact of network performance at the packet level on an application's end-to-end performance.

In contrast, the end-system has a variety of data sources available (e.g. TCP round-trip-time estimators, per-flow bandwidth estimators), and can accurately assign flows to applications (even, potentially, to different sub-parts of applications). This enables a much richer set of queries to be posed and answered (e.g. 'what is the contribution of link l to the delay experienced by the VoIP calls between h_1 and h_2 ?').

Current research approaches to distributed network management address problems of wide area networks through distributed deployment of standard tools such as *traceroute* and *ping*. Anemone targets enterprise networks where routing data is available for recovery of topology, and where end-systems are tightly controlled so that the majority can be made to provide real-time flow data. Enterprise networks have not been widely studied in the research community, with previous work generally restricted to either university campus or ISP networks. In short, an enterprise network will be similar in size to a large ISP network but will run many more services and be managed by a smaller, less well supported team with a greater degree of control over end-systems. In contrast to a university campus network, it will be larger, more geographically spread, but often better supported and the end-systems more homogeneous.

In more detail, a large ISP network will contain hundreds or thousands of routers spread across continents and split between backbone routers (for core connectivity) and data-centre routers (for aggregating customer access links and routing within data-centres). It will have multiple connections to multiple other networks, and deal with a customer base that generates

hundreds of thousands of flows per minute. To manage all this, there will be a dedicated team of skilled network operators, controlling the network from a central NOC (Network Operations Centre) using a variety of commercial and in-house tools. The major service offered will be simple connectivity, and basic services such as naming, email, web hosting and web proxy.

Conversely, a university campus network might span tens of sites in a single city, and support tens of thousands of users over 10/100 Ethernet. Services to support the entire user population will include email, web and file store. Campus networks tend not to be as tightly controlled in terms of end-systems as enterprise networks, and in terms of network access as ISP backbones, if only due to the relatively poorly supported and overly stretched operations staff.

Finally, a large enterprise network will contain hundreds or thousands of routers spread over hundreds of sites supporting hundreds of thousands of end-systems. Major sites will have multiple redundant connections to the backbone, minor sites may have only one. Additionally, many users may be supported by remote access across the public Internet, using tunnelled protocols such as PPTP. The set of services supported will be large, including email, directory, naming, web proxy, file storage and perhaps voice-over-IP. However, the operators will have a very high degree of control over the software and even usage of end-systems accessing the network.

The sheer scale of a large enterprise network, containing on the order of 10^5 hosts and 10^3 routers, clearly makes it impractical to collect flow data from every single end-system. A significant challenge of our approach is determining the required proportion of hosts to instrument in order to give acceptable network coverage and accuracy in query results. The problem is analogous to that of sampling packets in order to infer flow volumes, but we claim that we can exploit the asymmetric nature of enterprise network traffic patterns and topologies to overcome incomplete monitoring coverage. We outline the design of Anemone and discuss benefits and challenges in Section 2, followed by our prototype implementation in Section 3. We present an initial exploration of this issue through simulation in Section 4, before briefly reviewing related work in Section 5 and concluding in Section 6.

2 Overview, applications, challenges

The Anemone platform comprises 3 main components, depicted in Figure 1:

- End-system instrumentation recording locally transmitted and received flows together with the associated service or application.

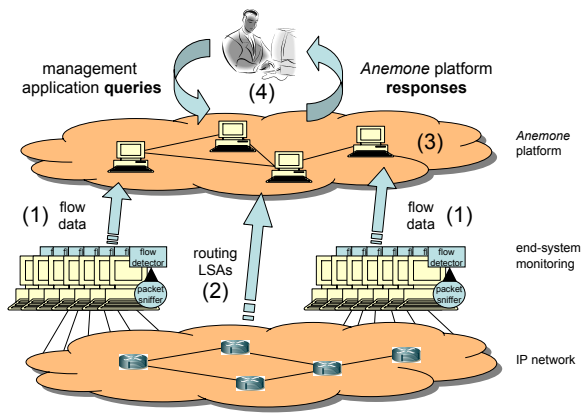


Figure 1. Anemone system architecture: (1) end-system instrumentation provides traffic flow statistics, (2) routing protocol monitoring provides current topology data, for (3) a data mining platform to (4) respond to queries posed by applications.

- A passive routing protocol monitor that listens to the link state advertisements and reconstructs the current routing topology.
- A data mining platform that combines these two datasets (flow data and topology), and provides APIs that support the queries generated by a variety of network management applications such as simulation, visualization and planning tools.

The core of Anemone is to leverage control of the end-system to monitor traffic entering and exiting the network, and to combine this with knowledge of the dynamic topology gathered by passive monitoring of routing protocols. This contrasts with current approaches which rely heavily on extensive device support to monitor from the core of the network. The combination of these two data sources will form a platform on which to build network tools providing features such as:

Visualization and logging, giving real-time and historical views of network topology and current traffic distribution on that topology. Such a tool would be invaluable for network monitoring and management, and for providing facilities such as traceback (allowing packets to be traced back to their entry points), and anomaly detection (allowing worms, denial-of-service and other malicious behaviour to be detected through the anomalous traffic patterns they generate).

For example, a real-time topology graph of the network with links colour-coded by load could make it straightforward to determine the target and entry-points of traffic causing distributed denial-of-service attacks. Misconfigured servers, routers, and infected machines all generate peculiar traffic patterns which could potentially be tracked through this system [13, 15].

Analysis and simulation, enabling ‘what-if’ analysis of the network to investigate and *predict* potential

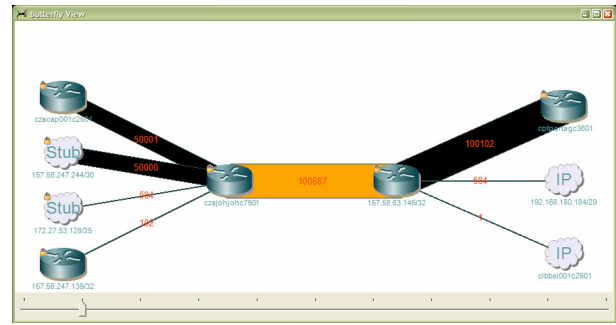


Figure 2. Screenshot from sample application built over Anemone. The link of interest is highlighted in the centre; on the left are the links that are currently feeding traffic into the link of interest, and on the right are the links that are being fed by the link of interest. All links are annotated with their current load.

changes in topology, configuration, and traffic distribution. This would be a powerful tool for answering questions such as that posed above (“what happens to the network if we consolidate all the mail servers?”), or concerning network response to failures or planning decisions.

For example, by feeding live flow and topology information into a flow-optimization solver the current network configuration could be evaluated for efficiency and robustness. Additionally proposed network modifications or potential failure scenarios could also be tested against current flow statistics in near real-time and hence over a period of days or weeks, avoiding reliance on sampled statistics such as ‘busy hours.’

Traffic engineering, where network configuration is automatically modified to control the flow of traffic through the network to meet imposed cross-network latency targets, link utilization targets, and so on. This would allow high-level policies requiring dynamic network reconfiguration to be applied automatically and corrected as the network configuration changes.

For example, the network could be dynamically re-configured as capacity is added so that service level agreements are always met in the most efficient way possible. It might also be possible to actively respond to detected traffic anomalies to reconfigure the network to contain the effects of malicious traffic.

A simple GUI application built over our simulator using our APIs helps demonstrate the power of Anemone. Figure 2 contains a screenshot giving a “butterfly view” of a backbone link indicating the load on the highlighted link and showing the respective contributions (the left hand “wing”) and withdrawals (the right hand “wing”) by each of the links connected to the highlighted link. This kind of flow-level data would be difficult to infer from traditional core-based network monitoring systems.

Challenges

Although this seems an appropriate architecture for a network management system, there are some research challenges that must be addressed in the process of its construction. These can be divided into five main areas, discussed below.

Engineering. There are two obvious challenges to be addressed here: multi-homed hosts; and ‘corner cases’ such as L4 routing, NAT, and transparent proxies. The first simply requires appropriate definition of a *flow*; as discussed in Section 3.1 this definition is left to run-time so there is no reason that this need be a problem. The second is somewhat more complex to address since all the techniques listed attempt to circumvent standard IP routing. This challenge should be addressable by appropriate use of SNMP to access MIB information about the configuration of devices implementing such features.

Scalability. This is purely an efficiency issue: is it possible to implement the monitoring and aggregation functions so that there is negligible impact on the packet latency, CPU load, and consumed network bandwidth at hosts and servers? Related work on distributed monitoring and the initial feasibility data presented in Section 4 suggest that it will be possible to implement the monitoring function suitably efficiently. Similarly, related work concerning highly scalable distributed databases such as SDIMS [28] suggest that the same can be said of the aggregation functionality.

Security. The threat of malicious/compromised hosts introducing false data into the system can be addressed in two ways: authenticate system participants, and cross-check reported data to detect liars. Since routing data is collected passively, no holes are introduced into the network’s routing system. Eavesdropping similarly requires standard encryption solutions perhaps based around IPsec. Finally, problems of denial-of-service on monitors/aggregators will probably be best dealt with by making the system self-tuning, for example adaptively rate pacing output from monitors when they become too busy.

Robustness. There are two aspects to robustness in this system: how node failure and network partition are handled, and how late and partial data (due either to unmonitored end-systems or to intentional aggregation) are handled. The first should be addressed through the peer-to-peer properties of the aggregation system. The second is an open research question: at least one aspect is the tension between the accuracy and timeliness of monitored data, and the volume of data in the system.

Control. To achieve the final goal of the system, automated network control, three questions must be answered: (i) is it worth doing (how efficient is the current configuration)? (ii) what are the feasible

timescales of reconfiguration, particularly given the tension mentioned above between accuracy/timeliness and volume of data? and (iii) how can network devices best be remotely reconfigured (a notoriously involved and fragile process)? The first two are open questions; the third has been addressed in the past, with solutions ranging from standardised control protocols, to the less robust but more widely supported remote invocation of the device’s command line interface.

3 Components and prototypes

We now describe our prototype implementations of the three components (the end-system instrumentation, passive routing protocol collection, and distributed data mining platform).

3.1 End-system instrumentation

We prototyped our end-system instrumentation over Microsoft Windows by modifying the simple *passthru* driver sample from the Windows Driver Development Kit to act as an ETW (Event Tracing for Windows) [18] event producer. Events are posted whenever packets enter and leave the network stack, and contain the usual IP 5-tuple¹. A user-space process then consumes these events in real-time to synthesise the flow data. This provides the required data for Anemone, as well as allowing the instrumentation (which would ideally be within the kernel network stack itself) to remain agnostic to the current definition of flow (which may vary based on the query or application).

This approach gives a great deal of flexibility in our treatment of flows since we can modify the definition of a “flow” according to the traffic semantics of the application. For example, a VoIP application might comprise multiple IP 5-tuple flows (a number of RTP/RTCP flows, an initial RTSP flow), including flows that use dynamically negotiated source and destination ports. Our approach would allow all of these 5-tuple flows to be accounted back to the VoIP application as a single flow if a particular query required.

3.2 Passive topology discovery

Large enterprise IP networks typically run a link-state routing protocol to manage routes within the network as these protocols are widely supported, well-understood, and relatively easy to manage. By collecting the link-state information disseminated in such protocols, we enable Anemone to use the same topology as the routing algorithms in the routers. Additionally, and in contrast to topology discovery approaches that

¹(source address, destination address, protocol number, source port, destination port).

rely on SNMP, *traceroute*, or other probing techniques, we are automatically notified of changes in the topology on the same timescale as other routers. Routing protocol snooping provides a very lightweight mechanism for acquiring a view of network state that is both timely and correct.

Link-state routing protocols operate in four stages: (i) routers form one-hop adjacencies with neighbours to monitor the state of the connecting link; (ii) each router creates and floods an LSA (Link State Advertisement) containing summaries of its live adjacencies; (iii) each router constructs the current network topology by forming a link-state database containing the most recent LSAs received from all routers in the network; and finally (iv) each router runs a shortest-path algorithm over this database to determine routes to all destinations. The resulting set of shortest paths allows the appropriate output port to be selected for any incoming packet. Although we assume the use of OSPF [20], a common IP link-state routing protocol in the remainder of this section, the same techniques should apply to other link-state routing protocols.

Access control limitations in our corporate network require that our routing protocol snooping infrastructure is *completely* passive. As LSAs are flooded to all local routers and our monitor sits on an Ethernet that connects two of the routers, we collect the LSAs describing our local area by simply putting our NIC into promiscuous mode. OSPF refresh timers guarantee that we recover a complete network topology within 30 mins of starting snooping. However, gaining access to the backbone (in fact, any non-local) area LSAs in a similarly passive manner is less straightforward since non-local routers do not learn of each other but only of each other’s connected networks, through various summarization mechanisms not detailed here. With the co-operation of our operations staff, we have a GRE tunnel configured between a backbone router and a (non-forwarding) local router. We physically route this tunnel through a local Ethernet switch and enable port spanning from the tunnel’s switch port to our passive monitor’s switch port. This allows us to passively monitor all the non-local area’s LSAs without requiring administrative access to any infrastructure network equipment, removing a potential network security hole².

In common with many larger enterprise networks, our network also deploys BGP (Border Gateway Protocol v4) internally to overcome the scaling limitations of the link-state protocol. We have not yet deployed BGP data collection as part of Anemone, although tools are available [19].

²Many thanks to our network operations staff for designing and implementing this!

3.3 Data mining platform

The tension in designing the data mining platform is between efficient resource usage and the robustness and ease of management of the data store. Logically, Anemone first aggregates and combines flow information collected by each end-system to construct the complete traffic matrix, $A_{ij} = \{\text{bandwidth from src } i \text{ to dst } j\}$, annotates each entry, (a_{ij}) , with the route from i to j , and finally executes queries supplied by applications against this dataset. Distributed database systems, such as those listed at the end of Section 5, augmented by the ability to perform the necessary route computation could provide a sound basis for the data mining platform in Anemone, making the annotated traffic matrix available to multiple querying end-systems in a robust, efficient, scalable fashion.

The development of our data mining platform is currently work in progress. We have a simulation of the platform which constructs a centralized database containing the augmented traffic matrix, and exposes a simple API allowing the database to be queried. The database is populated with real topology data from the OSPF listener on our network and synthetic traffic traces. This permits us to explore various design decisions and trade-offs concerning the degree of data distribution and aggregation required, some of the communication overheads of the platform, and the nature of the APIs provided to query the platform.

Data distribution among nodes will depend very much on the characteristics of the data and on the queries executed on the data. Given the datasets we have collected so far, it appears *a priori* reasonable to distribute the relatively static topology data to all (or most) end-systems, where it can be locally combined with the much more dynamic flow data that each end-system collects.

We are also using our simulator to explore the new types of network management queries we could support and the most efficient way to compute them. Currently we are studying basic queries including ‘what is the load on link l ?’, ‘what is the load {forwarded,sourced,sunk} at router r ?’, ‘which are the top- N busy links?’. A number of these queries utilize an optimization made possible by knowing the network topology: the predecessor matrix (an output of the Dijkstra computation) allows the set of hosts that might possibly be using a link to be pre-computed, reducing the communication overhead required by such queries.

4 Initial results

We have begun to investigate various system trade-offs and performance issues using our prototypes, simple packet tracing, and our simulator. Considering first the

basic overheads of end-system flow monitoring, we note that published figures for ETW state it is low overhead, able to post $\sim 20,000$ events per second (enough for an event every context switch) at an overhead of $< 5\%$ [1]. Packet traces taken over a 24 hour period from a typical client and server suggest that the number of flows to be monitored at any given moment is small: the client had < 39 flows active in any given second, and the server < 567 . State requirements are similarly manageable: if a flow timeout of 10 mins is used (excessively large for nearly all applications!) the client had < 567 concurrent flows, and the server < 5657 .

Next we consider the requirements of the topology dataset. OSPF data collected from our local OSPF area and from a backbone OSPF area show that it is typically small and quite stable. The local area LSA database contained 700 LSAs totalling 21kB, and the backbone area database contained 1048 LSAs totalling 34kB. Over a month, both areas experienced events (link failure, recovery, or reconfiguration) at an average rate of < 7 per hour; events tended to occur in bursts separated by days of stability.

One key short-coming of our current simulation setup is the lack of good models of traffic characteristics in enterprise networks: we are not aware of any such public models. The results below use a model based on packet data collected from a typical machine in our network; we are in the process of collecting more data to construct a more complete traffic model.

The model we use is very straightforward, capturing flow inter-arrival time (currently a simple Poisson arrivals process), flow size (a heavy-tailed distribution as expected), and flow transmission rate. We couple this with the OSPF data we have collected to incorporate a simple notion of the distribution of end-points of a flow (whether, given a flow’s source, its destination is within the subnet, the site, the area, the AS, or external). Using this model, we synthesized multiple 1 hr simulated traces placing 2000 hosts in the given topology and varying the proportion of instrumented hosts between 10–100%. The instrumented hosts were selected at random from the complete set of 2000 hosts.

Figure 3 gives a measure of system accuracy over the entire simulation run. We define the *link coverage* of a given link to be the proportion of the traffic on that link that the system is aware of at a given time, and the *system coverage* as the link coverage averaged across all links throughout the simulation. For each simulation we plot the proportion of instrumented hosts against the system coverage, as well as the (min,max) values obtained for the link coverage during the simulation. These results show that the accuracy of the system does not depend linearly on the proportion of instrumented hosts: unsurprisingly, which hosts in particular are instrumented (equivalently, the topology of the network)

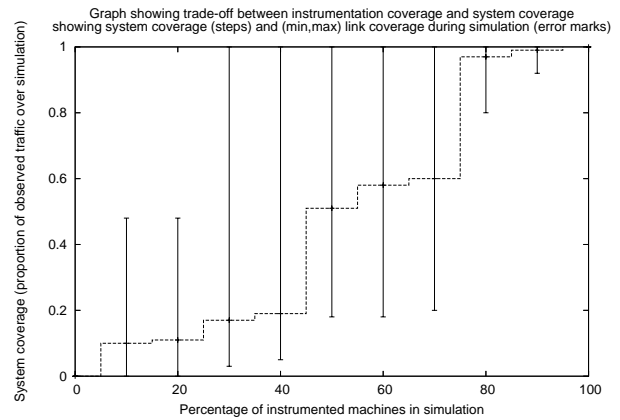


Figure 3. Graph showing the effect on the system coverage of varying the proportion of instrumented machines. Steps show system coverage for a given proportion of instrumented machines, with error bars showing the minimum and maximum link coverage values achieved throughout each 1 hour simulation.

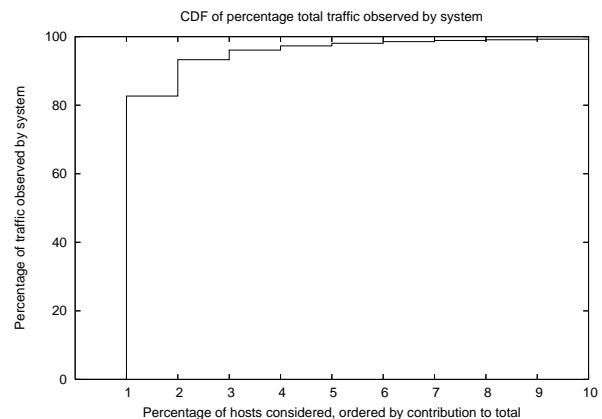


Figure 4. Plot showing the relationship between the proportion of instrumented machines and the total traffic observed by the system over a 1 hour simulation run, if the best possible set of machines is instrumented at each simulator tick.

and the traffic model combine to make this relationship quite non-linear.

This non-linear relationship is shown more clearly in Figure 4. This shows the accuracy of the system as it considers progressively more instrumented hosts, preferentially selecting the most significant contributors from the instrumented hosts at each simulator tick. This scenario is unrealistic but serves as a target against which performance can be measured. The principle takeaway here is that, at any given moment, perhaps 5% of hosts are observing over 97% of the traffic.

Finally, Figure 5 supports anecdotal evidence that in enterprise networks (and others!) servers account for the vast majority of the traffic, and a small number of servers account for most of that. Using a 24 hour packet trace containing inter-VLAN and WAN traffic originating on our LAN, it shows that 40 hosts, 16 of

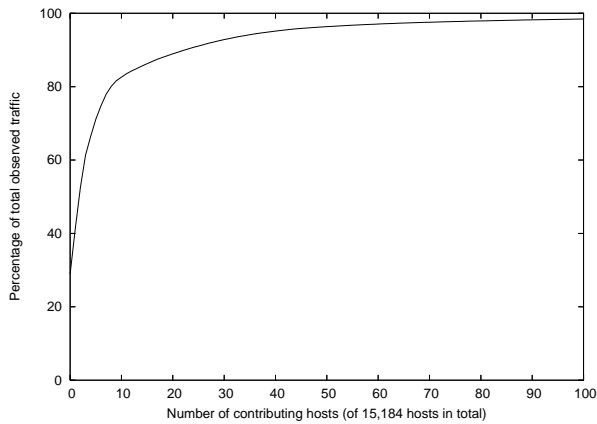


Figure 5. Plot showing the CDF of the percentage of all bytes observed per-host in a 24 hour trace taken on our LAN. 447.5 GB of IP traffic was observed during the trace, involving 15,184 hosts in total (transmitters and receivers – not all belong to our LAN!).

which are servers, observe 95% of this traffic. The top 5 hosts, all servers residing on our LAN, account for 66% of the traffic. The trace contained a total of 447.5 GB of transmitted traffic, and reference to 15,184 transmitting or receiving hosts. We believe that this high degree of asymmetry is typical of enterprise networks, and thus careful selection of hosts to instrument (i.e. instrumenting the servers) should allow us to achieve high coverage using only a small percentage of instrumented machines.

5 Related work

Network management has been an active research area for over 30 years as networking has moved from the PSTN telephone network through SONET, ATM, ISDN and broadband to IP networking and the wireless ad hoc networks of today. It is something of a catch-all field, covering performance management, configuration management, fault management, security management, and accounting. Essentially, the network management platform collects and presents information about the current status of the network to operators or for further application processing.

Historically, work in this arena has addressed two major issues. First, the definition and handling of management information for use by management applications. This involves issues around appropriate collection and presentation of data: filtering, storage, liveness, and so on, and includes major efforts by the IETF and ISO developing the standard MIBs for TCP/IP [3, 25, 17, 2]. Internet MIBs store information such as the IP addresses the router has observed as active, per-port byte and packet counters, and general configuration information. Traps might notify that a particular counter had exceeded a set limit.

Second, the design of automated or adaptive management systems, which utilise the data stored and presented by the management information base to control the system. Examples include the use of forward and backward inference for prediction and diagnosis in ATM networks [16], declarative logic rules applied to an object oriented database network model [7, 27], and the combination of declarative logic with active and temporal databases [11, 26]. The ETSI and ITU also had a long-running effort developing the notions of the TMN (Telecommunications Management Network) and subsequently the IN (Intelligent Network) [9, 14].

The work of the IETF and ISO developing MIBs was largely successful. Current IP network management products make extensive use of device MIBs, using ICMP Echo (ping) [22] for initial device discovery, and then SNMP (Simple Network Management Protocol) [2] to *get/set* MIB entries and to allow devices to asynchronously trigger actions in listening management systems via traps. Cisco routers also support NetFlow [4], a built-in measurement system able to present to management systems data concerning the current traffic at a router.

Unfortunately, none of these systems is really satisfactory: they require extensive and correct MIB support in the tool and on the device; they tend not to scale well, generating large volumes of data where core network solutions such as NetFlow are deployed; and they typically cannot provide an accurate, detailed view of network behaviour, due to the significant CPU and network load that frequent SNMP polling generates. NetFlow also suffers from the limitation that in the core it must typically use sampling techniques to monitor flows, giving rise to sampling artefacts and limiting its maximum accuracy in the face of short lived flows [6]. Even where these tools are successfully deployed, they do not address some of the fundamental issues raised above concerning the dynamic behaviour of the network.

More recently, distributed network monitoring has received much attention from the research community. Projects such as Neti@Home[21], ForNet [23], DIMES [5] and Domino [29] all use agents on end-systems to monitor network traffic, whether for intrusion detection and response or for straightforward network mapping and performance. The essential difference between these approaches and the approach taken in this paper is that they address problems of wide area networks through distributed deployment of standard tools such as *traceroute* and *ping*. In contrast, the system in this paper is targeted at enterprise networks where routing data is available for recovery of topology, and where end-systems are tightly controlled so that the majority can be made to provide real-time flow data.

Distributed databases supporting network management, from MANDATE [10] to the more recent peer-to-

peer inspired Astrolabe [24], PIER [12] and SDIMS [28], address database design specifically for storage and manipulation of network management data. They consider the need for scalability and distribution of the database, the inappropriateness of the ACID property, the requirement to support active database/trigger mechanisms, and the temporal nature of the data involved. Section 3.3 considers the potential of such systems to form the core of the platform.

6 Summary and future work

We have described *Anemone*, an end-system platform for network management. Anemone uses end-systems as real-time ‘network sensors’ to collect data about the network’s topology, and traffic in terms of flows. These datasets permit a variety of sophisticated network management queries to be answered, enabling network operators to regain control of their networks in the face of protocols that are opaque to the core of the network (e.g. IPSec, tunnelling). Additionally, Anemone allows the end-to-end impact of the network on application performance to be better understood, and through the use of space disk capacity in end-systems, enables the evolution of network usage to be tracked. We are currently further developing our simulation setup, at the same time as completing design and implementation of the three main components.

References

- [1] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. In *6th Symposium on Operating Systems Design and Implementation (OSDI’04)*, Dec. 2004.
- [2] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. Introduction to version 2 of the Internet-standard Network Management Framework. RFC 1441, IETF, Apr. 1993.
- [3] V. Cerf. IAB recommendations for the development of Internet network management standards. RFC 1052, IETF, Apr. 1988.
- [4] B. Claise. Cisco Systems NetFlow Services Export Version 9. RFC 3954, IETF, Oct. 2004.
- [5] Dimes project. <http://www.netdimes.org/>, Aug. 2004.
- [6] N. Duffield, C. Lund, and M. Thorup. Estimating flow distributions from sampled flow statistics. *Computer Communication Review (CCR)*, 33(4):325–336, Oct. 2003. Proceedings of ACM SIGCOMM 2003.
- [7] A. Dupuy, S. Sengupta, O. Wolfson, and Y. Yemini. NET-MATE: A network management environment. *IEEE Network Magazine*, pages 35–43, Oct. 1991.
- [8] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better NetFlow. In *Proceedings of ACM SIGCOMM 2004*, Portland, OR, Aug. 2004.
- [9] R. Glitho and S. Hayes. Telecommunications management network: Vision vs. reality. *IEEE Communications Magazine*, 33(3):47–52, Mar. 1995.
- [10] J. R. Haritsa, M. O. Ball, N. Roussopoulos, A. Datta, and J. S. Baras. MANDATE: MANaging Networks using DATabase TEchnology. *IEEE Journal of Selected Areas in Communications*, 11(9):1360–1372, 1993.
- [11] M. Hasan. An active temporal model for network management databases. In *Proceedings of the IFIP/IEEE Fourth International Symposium on Integrated Network Management*, pages 524–535, Santa Barbara, CA, May 1995.
- [12] R. Huebsch, J. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *29th International Conference on Very Large Data Bases (VLDB ’03)*, Sept. 2003.
- [13] A. Hussain, J. Heidemann, and C. Papadopoulos. A framework for classifying denial of service attacks. *Computer Communication Review (CCR)*, 33(4):99–110, Oct. 2003.
- [14] M. Kockelmans and E. de Jong. Overview of IN and TMN harmonization. *IEEE Communications Magazine*, 33(3):62–66, Mar. 1995.
- [15] A. Lazarevic, L. Ertoz, A. Ozgur, J. Srivastava, and V. Kumar. A comparative study of anomaly detection schemes in network intrusion detection. In *Proceedings of Third SIAM Conference on Data Mining*, San Francisco, CA, May 2003.
- [16] Y.-D. Lin and M. Gerla. Induction and deduction for autonomous networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 11(9):1415–1425, Dec. 1993.
- [17] K. McCloghrie and M. Rose. Management Information Base for Network Management of TCP/IP-based internets:MIB-II. RFC 1213, IETF, Mar. 1991.
- [18] Microsoft. Event tracing. <http://msdn.microsoft.com/library/>, 2002. Platform SDK: Performance Monitoring, Event Tracing.
- [19] R. Mortier. Python routing toolkit. *IEEE Network Magazine*, 16(5):3–3, Sept. 2002. Appeared in the ‘Software tools for networking’ column.
- [20] J. Moy. OSPF Version 2. RFC 2328, IETF, Apr. 1998.
- [21] Neti@home project. <http://www.neti.gatech.edu/>, Aug. 2004.
- [22] J. Postel. Internet Control Message Protocol. RFC 0792, IETF, Sept. 1981.
- [23] K. Shanmugasundaram, N. Memon, A. Savant, and H. Bronnimann. ForNet: A distributed forensics network. In *Proceedings of the Second International Workshop on Mathematical Methods, Models and Architectures for Computer Networks Security*, St. Petersburg, Russia, 2003.
- [24] R. Van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems (TOCS)*, 21(2):164–206, 2003.
- [25] U. Warrior, L. Besaw, L. LaBarre, and B. Handspicker. Common Management Information Services and Protocols for the Internet (CMOT and CMIP). RFC 1189, IETF, Oct. 1990.
- [26] M. Wawrzoniak, L. Peterson, and T. Roscoe. Sophia: An information plane for networked systems. In *Proceedings of the Second Workshop on Hot Topics in Networking (HotNets-II)*, Cambridge, MA, Nov. 2003.
- [27] O. Wolfson, S. Sengupta, and Y. Yemini. Managing communication networks by monitoring databases. *IEEE Trans. Softw. Eng.*, 17(9):944–953, 1991.
- [28] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *Proceedings of ACM SIGCOMM 2004*, Sept. 2004.
- [29] V. Yegneswaran, P. Barford, and S. Jha. Global intrusion detection in the DOMINO overlay system. In *Proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS)*, Feb. 2004.