Reclaiming Network-wide Visibility Using Ubiquitous Endsystem Monitors

Evan Cooke
University of Michigan
`emcooke@umich.edu`

Richard Mortier, Austin Donnelly, Paul Barham, Rebecca Isaacs
Microsoft Research, Cambridge
{`mort,austind,pbar,risaacs`}`@microsoft.com`

Abstract

Network-centric tools like NetFlow and security systems like IDSes provide essential data about the availability, reliability, and security of network devices and applications. However, the increased use of encryption and tunnelling has reduced the visibility of monitoring applications into packet headers and payloads (e.g. 93% of traffic on our enterprise network is IPSec encapsulated). The result is the inability to collect the required information using network-only measurements. To regain the lost visibility we propose that measurement systems *must* themselves apply the end-to-end principle: only endsystems can correctly attach semantics to traffic they send and receive. We present such an end-to-end monitoring platform that ubiquitously records per-flow data and then we show that this approach is feasible and practical using data from our enterprise network.

1 Introduction

Network enabled applications are critical to the running of large organisation. This places great importance on monitoring methods that provide visibility into the network. Tools such as NetFlow [4] are becoming essential for efficient network management, and continuous network monitoring platforms are the subject of ongoing research [8, 12]. However, these tools assume that in-network tools have direct access to packet headers and payloads. Unfortunately, the composition of network traffic is changing in ways that directly impact our ability to solve security and network availability problems.

Two trends are particularly troubling for monitoring approaches relying on a network-centric perspective. First, the use of encryption and tunnelling preclude any inspection of payloads; use of tunnelling may also prevent connection information from being determined. Second, the complexity of network applications means that tracking network application behaviour can require visibility into traffic to many destinations using protocols and ports that are expensive to differentiate from other traffic [9]. This is particularly notable in modern enterprise networks where many services are provided over dynamically allocated ports. As a result, collecting all the packets at an upstream router is often no longer sufficient to report on the performance or diagnose problems of downstream network applications. Essentially, network operators have contradictory requirements: security 'in depth' through mechanisms leading to opaque traffic (e.g. IPSec) *and* fine-grained auditing only available through traffic inspection.

We propose to provide this fine-grained auditing capability without restricting the ability to deploy essential security mechanisms by using information collected on endsystems to reconstruct an 'end-to-end' view of the network. Each endsystem in a network runs a small dæmon that uses spare disk capacity to log network activity. Each desktop, laptop and server stores summaries of all network traffic it sends or receives. A network operator or management application can query some or all endsystems, asking questions about the availability, reachability, and performance of network resources and servers throughout the organization. We initially target deployment in government or enterprise networks since these exercise a high degree of control over endsystems. This makes it feasible to deploy a standard operating system image supporting the monitoring facility, and to control data logging in a manner consistent with network security and privacy policies.

Ubiquitous network monitoring using endsystems is fundamentally different from other edge-based monitoring: the goal is to passively record summaries of *every* flow on the network rather than to collect availability and performance statistics or actively probe the network. Projects such as DIMES [5] and Neti@Home[14] use endsystem agents to monitor network properties (e.g. availability and reachability). The Anemone system also collects endsystem data but combines it with routing data to construct a view of the network [13].

In contrast, ubiquitous endsystem network monitoring is more closely related to in-network monitoring approaches like NetFlow [4] in that it provides summaries of all traffic on a network. It also provides a far more detailed view of traffic because endsystems can associate network activity with host context such as the application and user that sent a packet. This approach restores much of the lost visibility and enables new applications such as network auditing, better data centre management, capacity planning, network forensics, and anomaly detection. Using real data from an enterprise network we present preliminary results showing that instrumenting, collecting, and querying data from endsystems in a large network is both feasible and practical.

2 Muddy Waters

For many older network applications it is possible to reconstruct an entire application's session simply by observing the packets between a client and a server (e.g. telnet). The problem is that we have lost much of this visibility due to new network application requirements and deployment practices. Some of this loss can be attributed to new security and privacy features that have rendered a significant amount of network traffic opaque. Furthermore, increasingly complex application communication behaviour can make attributing traffic to specific applications complex and impractical.

2.1 Opaque Traffic

As the Internet has grown to support critical transactions like the transfer of money, security and privacy requirements have come to the forefront of application development and deployment. For example, many large organisations support remote offices and home workers using encrypted VPNs to tunnel traffic back to a central location. Devices that want to monitor this traffic must be placed before or after these tunnels or have access to any keys required to decrypt the traffic.

Two situations arise as a result of this need to monitor network usage. If an organization outsources the running of their network, then they must turn over any keys to their service provider, trusting that the provider will not disclose highly sensitive information. Alternatively, if they keep their network in-house, they must increase its cost and complexity by setting up tens or hundreds of monitoring points as well as the infrastructure required to collect and process the resulting data. Analysing a snapshot of the configuration files from 534 routers in a large enterprise network, we found 193 separate sites that would require integration into such a system – a substantial investment.

Problems with opaque traffic also exist within each office site network. To increase security within their LANs, organisations may use mechanisms such as IPSec to provide authentication of application packets destined for internal addresses. This claim is supported by analysis of an 8-day packet trace collected at a remote office site in large enterprise network. The trace included $\simeq$14,000 host source addresses, $\simeq$600 of which were on-site. Over 93% of the collected packets were transport-mode IPSec, and so encapsulated in an ESP header and trailer. This observation differs significantly from previously reported traffic measurements in ISP and educational networks [9, 15], and highlights one of the significant differences between controlled environments like enterprise networks and more open networks: even monitoring devices sitting inside a local LAN may have very limited visibility into network traffic.

2.2 Complex Application Behaviour

Even when a packet is not opaque, it may still be impractical to extract application information and behaviour. Compatibility and security requirements often result in applications that tunnel traffic using common protocols such as HTTP and other transports over varying ports [2, 3].

Modern applications may also exhibit highly complex communication relationships. For example, even an apparently simple application such as email often no longer operates straightforwardly between a given client and server over a single protocol such as SMTP. Instead, "checking your mail" can require connections to many servers such as to an authentication server in order to obtain appropriate credentials, to a mail server to authenticate the user to a mailbox, and finally many other connections to download different headers, mails, and attachments. In addition, the mail application may concurrently be performing background tasks such as synchronising address books, and maintaining calendar alerts. In all, "checking your mail" can instantiate tens of connections to several servers, making it problematic to attribute the relevant traffic to a single application.

In summary, increasingly opaque network traffic and complex application behaviour introduce significant visibility problems for network-centric monitoring approaches. The question is how we can obtain insight into the network in the face of these visibility problems.

3 End-to-End Network Monitoring

To provide the necessary visibility into network traffic we propose an endsystem-based network management platform that uses information collected at the edge to construct a view of the network. Each endsystem in a network runs a small dæmon that uses spare disk capacity to log network activity. Each desktop, laptop and server stores summaries of all network traffic it sends or receives. A network operator or management application can query some or all endsystems, asking questions about the availability, reachability, and performance of network resources throughout an organization.

To validate an endsystem-based monitoring approach we constructed a prototype designed to be integrated into the standard operating system distribution for a large enterprise. We used the Windows built-in kernel monitoring ETW [10] facility to report socket creation/destruction and data transmission/reception. A user-space Windows service processed the ETW events periodically outputting summaries of network data to disk. We mapped each network invocation to a specific application on the endsystem using process identifier information in the common ETW header. Flow tuples were recorded according to the following schema: *(sip, dip, dport, sport, proto, process_name, PID, bytes, packets, timestamp)*. This enabled the system to find all the network data associated with complex applications such as Microsoft Outlook and to observe network packets before being encrypted in the VPN layer. A preliminary evaluation of the prototype is described in Section 5.

4 Novel Uses

The fresh perspective provided by an end-to-end monitoring platform enables a range of new network management applications. In contrast to existing monitoring techniques, the platform provides fine-grained usage information in multiple dimensions (host, user, application, virtual machine) as well as the capacity to store this data for significant periods of time.

Network auditing. An operator wishing to determine who is using an expensive WAN link can query the system to determine all the hosts, applications, and users that have used the link over the past few weeks. The system could even be used in a feedback loop to throttle specific applications and hosts using more than their allocated share of a given network resource.

Data centre management. Fined-grained information on network usage can be attributed to individual applications, machines and even specific virtual machines. For example, one might use such a system to account network usage to individual users on a server hosting multiple virtual machines multiplexing a single IP address.

Capacity planning. By using historical data on network usage by specific applications stored on many endsystems, detailed models of application network usage can be built. These can predict the impact of service changes such as distributing email servers among many sites or concentrating them in a few datacentres.

Anomaly detection. Distribution of historical network data across many endsystems also enables new applications that require detailed historical context. For example, models of normal behaviour can be constructed from this extensive distributed archive and deviations from past behaviour detected. An operator could ask for a detailed usage report of all abnormal applications across an enterprise.

5 Feasibility Study

Having argued the need for, and utility of, a monitoring system with better visibility, we now consider several key implementation and deployment issues: (*i*) where might an end-to-end monitoring system be deployed? (*ii*) how many endsystems must be instrumented? (*iii*) what data should be collected? (*iv*) how can that data accessed? (*v*) what is the performance impact on participating endsystems? and (*vi*) what are the security implications?

(*i*) System Deployment. One major challenge for an endsystem monitoring platform is how to instrument edge devices. In many large networks, access to network-connected elements is strictly controlled by a central organization. In particular, enterprise and government networks typically have infrastructure groups that generate and enforce policies that govern what machines can be connected to what networks and what software they must run. These highly controlled settings are the perfect environment for end-system monitoring because they are also often quite large and require tools that can provide visibility across a whole network. For example, our own enterprise network contains approximately 300,000 endsystems and 2,500 routers. While it is possible to construct an endsystem monitor in an academic or ISP network there are significant additional deployment challenges that must be addressed. Thus, we focus on deployment in enterprise and government networks that have control over software and a critical need for better network visibility. We discuss the security and privacy implications of collecting endsystem data later.

(*ii*) Deployment Coverage. Even under ideal circumstances there will inevitably be endsystems that simply cannot easily be instrumented, such as printers and other hardware running embedded software. Thus, a key factor in the success of this approach is obtaining good visibility without requiring instrumentation of all endsystems in a network. Even if complete instrumentation were possible, deployment becomes significantly more likely where incremental benefit can be observed.

If traffic were uniformly distributed between N endsystems, then each additional instrumented endsystem contributes another $1/N$th to the global view. This observation is initially discouraging as it suggests that a

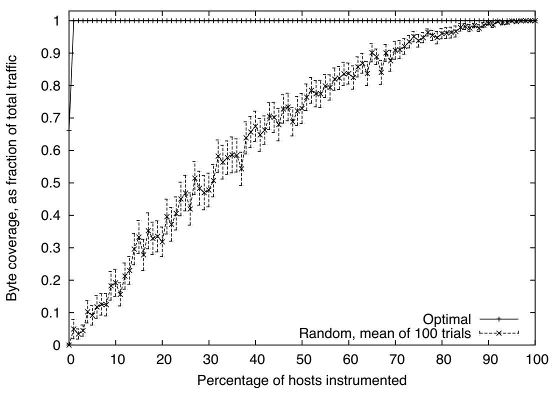

Figure 1: Fraction of traffic observed by endsystem monitoring as increasing subsets of endsystems are instrumented. Error bars on the *random* line show the 95% confidence intervals of the mean.

very high proportion of endsystems would have to participate to obtain a high percentage of traffic. However, if the traffic distribution is less symmetric, a smaller proportion of participating systems may still lead to a useful proportion of the traffic being observed.

To investigate the contribution of different endsystems to the overall traffic we analysed the 8-day network trace from a large enterprise network described in Section 2. We computed the number of bytes and flows observed by each endsystem for the entire trace and for different applications. We define the *byte coverage* as the proportion of the total number of bytes observed by the system across the network within a given time period, and the *flow coverage* similarly for network flows.

Figure 1 depicts the byte coverage as a function of the number of endsystems instrumented. The *optimal* line shows the coverage when endsystems were chosen based on their contribution to the total, largest first. This line shows that instrumenting just 1% of endsystems was enough to monitor 99.999% bytes on the network. This 1% is dominated by servers of various types (e.g. backup, file, email, proxies), common in such networks. Note that in these controlled networks, traffic to and from external addresses will typically traverse some kind of proxy device. With simple modifications to track the setup and teardown of connections for the proxied traffic, these devices can provide excellent visibility into traffic where one side of the connection is not itself part of the controlled network.

The *random* line in Figure 1 shows the mean and 95% confidence intervals across 100 trials when endsystems are selected at random from the total host population. In this case, the resulting coverage is slightly better than linear since both transmit and receive directions are monitored, but variance is quite high due to the fact that if a given trial includes just one of the top 1% endsystems, it leads to a disproportionate improvement in coverage. We also measured flow coverage since that is of interest for a number of security applications. The results were very similar to the byte coverage.

Finally, since per-application information is a significant benefit of our approach we also analyzed byte and flow coverage for the top 10 applications (for bytes transmitted and received). Again, the results were almost identical to Figure 1: because many enterprise applications are heavily client-server based, it is possible to achieve excellent visibility into them all with just a few instrumented machines. Even if enterprise network workloads become peer-to-peer dominated in the future, a significant shift away from centralized communication models, the worst case is that each extra machine instrumented brings only a $1/N$ improvement (for N the number of end-systems on the network and assuming a completely uniform traffic distribution). Since many centralized applications like security proxies are not suited to peer-to-peer topologies, it is likely that a balance of different topologies will likely persist.

***(iii)* Data Collection**. One common method of storing network data is to capture all packets using a packet sniffer, but this can result in unmanageably large datasets. For example, even a moderately busy server transmitting at 100 Mbps would result in recording gigabytes of data per hour if just the IP and TCP headers were recorded. Since the 1% of endsystems that provide the best coverage are often precisely the busiest 1% of endsystems, a more scalable approach is highly desirable.

The problem of collecting and storing data is well-known in network-centric monitoring. It is often infeasible for routers and other network devices to capture all the packets that they forward so they typically aggregate data, storing information about each *flow* rather than each packet (e.g. NetFlow [4]). Flow records provide excellent compression since a connection with hundreds of packets is synthesised into a single flow. The information in such a flow record might include timestamps, protocol, source, destination, number of packets in the flow, and other fields traditionally available through packet inspection such as TCP headers. Thus, using flows rather than packets provides nice tradeoff between resource cost and network information.

Flows can also be augmented with endsystem information. For example, the user executing the application, the current round-trip time estimates from the TCP stack. Furthermore, the monitoring software can be placed before encryption and tunneling layers so that the resulting flow records store both unobfuscated network activity with host contextual information. Application-level encryption such as SSL may require additional instrumentation of system libraries or applications.

***(iv)* Accessing Distributed Data Stores**. Assuming that we can instrument and collect flows on 300,000 different endsystems, the question becomes how to access

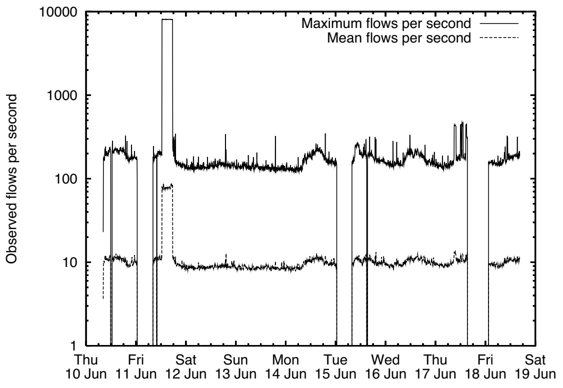

Figure 2: Maximum and mean observed flows per second. The three dropout periods occurred due to crashes of our monitoring system.

these widely distributed data stores in a timely manner. One approach is to collect and centrally store all flow records from all endsystems. This approach clearly doesn't scale well when more than a handful of busy endsystems are involved. Another possibility is to transfer summaries of flows to a central location. The problem is that because the total amount of data so large, deciding what to summarise requires network applications and operators to already know the questions they wish to ask. This is not always possible. For example, when examining the artifacts left by a network intruder.

A more flexible approach is to provide a platform allowing network management applications to insert queries and receive aggregated responses in real-time. Fortunately, a variety of distributed databases supporting access to network management data already exist [7, 16]. These systems have addressed scalability, the maintenance the ACID property, support for active database/trigger mechanisms, and the temporal nature of network data. By making each data collection node a member of a large distributed query system one could access data from across the entire system in a timely manner. For example, one could utilize the ability of SDIMS [16] to build dynamic aggregation trees or use hierarchical DHT rings [11] to multicast queries to specific sub-networks.

(*v*) Endsystem Performance. The busiest 1% of endsystems are often the most useful to instrument so an important concern is the impact of collecting and storing flows on an endsystem. In general, three key resources can be used: memory, CPU, and disk. Memory cost is a function of the number of concurrently active flows, CPU cost a function of of the number of packets observed, and disk cost a function of the number of records stored. We now estimate these costs using the packet trace described in Section 2.

Memory Cost. The memory cost on the host is dominated by state required to store flow records before they are exported to disk. To evaluate this cost we constructed

	Per-endsystem write rate	
Export Timer (s)	Mean (kB/s)	Maximum (kB/s)
1	1.445	2081.00
5	0.310	418.00
10	0.168	211.00
30	0.073	71.70
60	0.049	37.50
300	0.022	12.10
900	0.016	5.40
1800	0.016	5.40

Table 1: Per-endsystem maximum and mean rates at which records must be written for a variety of export periods.

a flow database for each of the endsystems. We denote a flow as a collection of packets with identical IP 5-tuple and no inter-packet gap of greater than 90 seconds. Figure 2 depicts the observed maximum and mean flows per second over all endsystems in the trace. Allowing a generous 256 bytes per in-memory flow record, the exceptional worst case memory consumption is under 2 MB, with an average of $\simeq$25 kB.

CPU Cost. To evaluate the per-endsystem CPU overhead we constructed a prototype flow capture system using the ETW event system [10]. ETW is a low overhead event posting infrastructure built into the Windows OS, and so a straightforward usage where an event is posted per-packet introduces overhead proportional to the number of packets per second processed by an endsystem. We computed observed packets per second over all hosts, and the peak was approximately 18,000 packets per second and the mean just 35 packets per second. At this rate of events, published figures for ETW [1] suggest an overhead of a no more than a few percent on a reasonably provisioned server.

Disk Cost. Finally, we consider disk cost by examining the number of flow records written to disk. Using the number of unique flows observed in a given export period as an estimate of the number of records that would need to be written, Table 1 shows the disk bandwidth required. For example, for a 1 second export period there are periods of high traffic volume requiring a large number of records be written out. However, if the export timer is set at 300 seconds, the worst case disk bandwidth required is $\simeq$4.5 MB in 300 seconds, an average rate of 12 kBps. The maximum storage required by a single machine for an entire week of records is $\simeq$1.5 GB, and the average storage just $\simeq$64 kB. Given the capacity and cost of modern hard disks, these results indicate very low resource overhead.

Initial results from our prototype system are very promising. Costs are within acceptable limits for the handful of key systems required for excellent visibility, and well within limits for most users' endsystems. Moreover, the costs described are not fixed performance requirements and operators could be given the ability to adjust the level of resource usage. For example, one could add an additional sampling parameter to reduce the number of flows processed and stored or use adaptive flow sampling approaches to further reduce load [6].

6 Security/Privacy Implications

Data available through such a system could potentially support a number of interesting network security and forensics applications. However, several security and privacy questions arise when collecting sensitive network traffic on many endsystems. Specific requirements will depend heavily on the deployment environment, e.g. in a corporate environment there are often regulations governing the protection of data. We now highlight a few of the important issues and suggest how a basic cross-validation strategy can help handle them.

To maintain endsystem integrity and communication security, simple procedures such as privilege separation for software, and encryption of queries may be used. Queries should be authenticated by requiring that they be properly signed by a designated authority, ensuring malicious parties cannot easily discover information about the network to their benefit, and helping prevent malicious or naive users from instigating denial-of-service attacks by introducing many excessively complex or long-running queries. Nefarious insertion or removal of flow data can be detected by asking a host to report on a known quantity, and then validating with the other ends of the flows. For example, a suspicious host could be queried for the amount of data it transmitted and to whom, and the receivers on the same network queried to validate that they received the data that was sent.

Endsystem monitoring also provides additional privacy protection compared to other monitoring approaches. Since each endsystem logs only the data it sends or receives, a node never has access to data that it hasn't already observed. Furthermore, each organisation can customize the data that is logged based on specific endpoints and applications. This is a significant advantage over in-network monitoring solutions where it is difficult to apply privacy filters when the data is recorded requiring that data be scrubbed later. And, an endsystem monitoring solution enables selected highly trusted systems to have different privacy policies than other parts of the network.

7 Conclusion

We believe that network centric monitoring approaches will continue to lose visibility into the network as traffic becomes more opaque and complex. Rather than directly instrument the network, we propose an end-to-end monitoring platform that uses data collected on endsystems to construct a view of the network. Endsystems are able to provide significantly more visibility than network devices which lack critical host context. An end-to-end platform also enables many new applications like auditing of network resources, better data centre management, capacity planning, network forensics, and anomaly detection. Our preliminary results using real data from an enterprise network show that collecting and querying data from endsystems in a large network is both feasible and practical.

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] S. Baset and H. Schulzrinne. An analysis of the Skype peer-to-peer Internet telephony protocol. Technical Report CUCS-039-04, Columbia University, 2004.

[3] K. Borders and A. Prakash. Web tap: detecting covert web traffic. In *ACM Conference on Computer and Communications Security*, Oct. 2004.

[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] C. Estan, K. Keys, D. Moore, and G. Varghese. Building a better NetFlow. In *Proceedings of ACM SIGCOMM 2004*, Portland, OR, Aug. 2004.

[7] 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.

[8] G. Iannaccone, C. Diot, D. McAuley, A. Moore, I. Pratt, and L. Rizzo. The CoMo white paper. Technical Report IRC-TR-04-17, Intel Research, Sept. 2004.

[9] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. Blinc: Multilevel traffic classification in the dark. In *Proceedings of ACM SIGCOMM 2005*, Aug. 2005.

[10] Microsoft. Event tracing. `http://msdn.microsoft.com/library/`, 2002. Platform SDK: Performance Monitoring, Event Tracing.

[11] A. Mislove and P. Druschel. Providing administrative control and autonomy in structured peer-to-peer overlays. In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS04)*, Feb. 2004.

[12] A. Moore, J. Hall, E. Harris, C. Kreibech, and I. Pratt. Architecture of a network monitor. In *Proceedings of the Fourth Passive and Active Measurement Workshop (PAM 2003)*, Apr. 2003.

[13] R. Mortier, R. Isaacs, and P. Barham. Anemone: using end-systems as a rich network management platform. Technical Report MSR-TR-2005-62, Microsoft Research, Cambridge, 7, JJ Thomson Ave, Cambridge, CB3 0FB. UK., May 2005.

[14] Neti@home project. `http://www.neti.gatech.edu/`, Aug. 2004.

[15] S. Saroiu, P. Gummadi, R. Dunn, S. Gribble, and H. Levy. An analysis of Internet content delivery systems. In *5th Symposium on Operating Systems Design and Implementation (OSDI'02)*, Dec. 2002.

[16] P. Yalagandula and M. Dahlin. A scalable distributed information management system. In *Proceedings of ACM SIGCOMM 2004*, Sept. 2004.