Anemone: Edge-based network management

Richard Mortier, Rebecca Isaacs, Austin Donnelly, Paul Barham*
Microsoft Research, 7, JJ Thomson Ave, Cambridge, CB3 0FB, UK.
Tel. +44 (0)1223 479000. Fax. +44 (0)1223 479999.
{mort,risaacs,austind,pbar}@microsoft.com

***Abstract*— This proposal describes the *Anemone* project and a demonstration of the work so far. The project is developing an edge-based IP network management platform which utilises only information collected at the edges of the network, eschewing the need to collect data in the network core. Devoting a small fraction of hosts' idle cycles, disk space, and network bandwidth to network management allows inference of network-wide traffic patterns by synthesising end-system flow statistics with dynamic topology information obtained through passive snooping of IP routeing protocols. We claim that this approach will provide a more complete view of the network that supports sophisticated traffic engineering queries to supply the global statistics necessary to automate network control, and is future-proofed against increasing deployment of encrypting and tunnelling protocols.**

I. Introduction

The *Anemone* project is addressing the challenge of IP network management, and *enterprise* IP network management in particular. Large IP networks are highly complex, largely due to the conjunction of two factors: the dynamic nature of the IP routeing protocols, and the size of such networks. The former means that it takes time for network devices to reach a 'converged' state even in a completely static network; the latter means that events such as link failure, recovery and device reconfiguration occur continuously, making such a converged state practically unattainable in any case. The resulting complexity makes managing such networks expensive, requiring manual intervention by skilled human operators.

Existing management tools for large IP networks tend to ignore the global dynamics of network behaviour, concentrating on centrally unifying potentially conflicting data taken locally from individual network devices. They make it difficult for operators to perform a variety of useful tasks such as discovering the path a particular set of traffic takes through the network, investigating the behaviour of the network in 'what-if' scenarios, or monitoring the evolution of the network as failures and recoveries occur. This situation is likely to become worse in the future, as protocols like IPSec or PPTP become widely deployed: these make traffic opaque to network devices, making untenable traditional approaches such as inspecting port numbers to infer application traffic distributions.

Even where such tools are available[1], operators will often fall back on a variety of *ad hoc* approaches using *traceroute*, *ping*, and so on. Indeed, much research work concerning network performance, reliability, and provisioning [7], [5], [8][2] relies entirely on a variety of *ad hoc* edge-based approaches. The general goal of the *Anemone* system is to bring some coherence to these approaches; the specific goal is to implement the system in an enterprise network context.

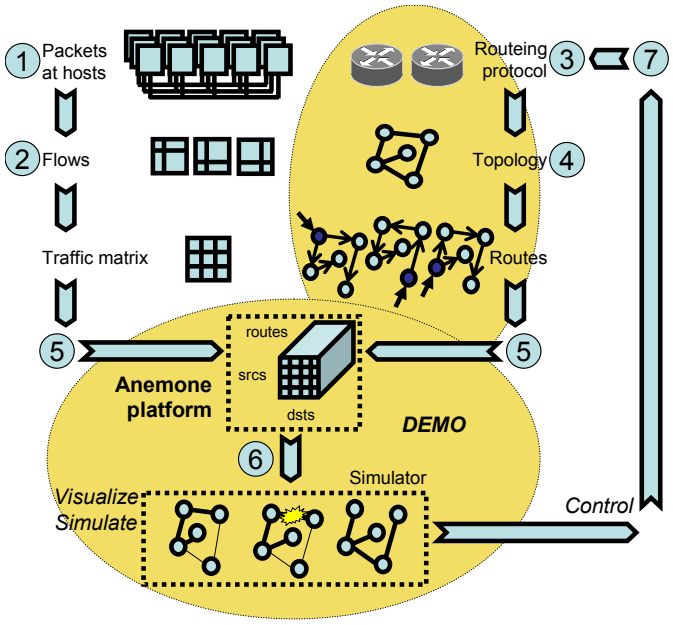

Fig. 1. Diagram of logical system architecture. On the left, packet activity is monitored at end-systems (1) to recover flows (2). On the right, the link-state routeing protocol is passively monitored (3) to recover the current topology (4). These two data sets contain the traffic matrix and the current routes from sources to destinations. They are then combined and aggregated in a distributed database (5), to be queried by applications for visualization and simulation (6). Finally, results from simulation or other prediction process can be fed back into the routeing protocol to control the network (7).

As a concrete example, consider an IT manager asked to consolidate their company's email servers at a single site. Very few tools exist to help them predict the impact on their network and the necessary reconfiguration resulting from changed traffic patterns. Most likely they would have to build *ad hoc* simulations of their network using generic traffic and failure distributions, possibly estimating parameters from measurement samples if these were available.

II. Anemone

The system architecture is shown in Figure 1. The core contribution of the project 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 passively monitoring routeing protocols. The combined data set (effectively, the traffic matrix, annotated with source-destination routes) can then be queried, forming a

* Corresponding author.

[1]And their prohibitive cost often prevents their purchase and installation.

[2]See also, for example, the ACM SIGCOMM Internet Measurement Workshops and Network Troubleshooting Workshops, `http://www.sigcomm.org/`.

platform for building network management applications. This contrasts with current approaches which rely on extensive device support in the core of the network. The *Anemone* platform will combine these two data sources to support 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 also enable facilities such as traceback (allowing traffic flows to be traced back to their entry points), and anomaly detection (allowing detection of worms, denial-of-service and other malicious behaviour through the anomalous traffic patterns they generate). Misconfigured servers, routers, and infected machines all generate peculiar traffic patterns which could be tracked through this system [3], [4].

Analysis and simulation, enabling 'what-if' analysis of the network to investigate and *predict* potential 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 reconfigured 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.

III. Benefits

A network management platform that unifies flow data from the end-systems and topology data from the routeing protocol appears to have three main benefits over traditional SNMP-based solutions: (*i*) ability to inspect tunnelled/encrypted protocols, widely deployed in IP networks and especially enterprise IP networks, (*ii*) access to the plentiful resources at the network edge in contrast to core routers which are often resource starved, and (*iii*) lack of reliance on particular features of or support from the many makes and models of device found in a modern network.

The first of these benefits is perhaps the most important. It arises due to the requirement that enterprise networks cheaply support many remote sites while remaining secure, giving rise to use of protocols such as IPSec and PPTP. These protocols prevent network devices easily examining and managing traffic from the centre of the network; in contrast the end-system has all the information required to decrypt and de-encapsulate traffic it transmits and receives. Similarly, the end-system is capable of directly tieing traffic to application without recourse to port numbers and other inference techniques.

The second arises because core routers must deal with processing high packet volumes and the many protocols that impact router behaviour (not least the routeing protocols!), while often having older CPUs which lack sufficient processing power. As a result, solutions such as NetFlow must often be deployed in a sampling mode due to the CPU cost at the monitoring router and the load generated by the NetFlow measurements themselves [1].

Finally, modern networks tend to be evolutionary in nature: it is difficult to perform complete network-wide upgrades of hardware due to the significant disruption in service this would cause. An end-system based approach means that heterogeneity of network devices is no longer a problem: no device manufacturer support is required beyond correct implementations of the routeing protocol and IP packet forwarding. Since these are critical for correct functioning of any router device, they are likely to be less buggy than non-critical SNMP MIB implementations. In fact, it is not necessary for devices within this system to support SNMP at all!

Furthermore, the structure of the proposed system has two additional benefits. First, decoupling basic traffic monitoring from flow accounting permits the definition of flow to be modifiable without requiring widespread software deployment each time it changes. Second, decoupling the data collection from the query system permits the use of techniques from semi-structured databases such as PIER [2] or SOPHIA [9] providing a genuinely flexible platform on which to build applications.

IV. Status

We are in the process of prototyping and simulating the system described in Section II. We have prototypes of the flow monitors and route collectors, and we are simulating the platform itself using route data collected using our prototype and flow data generated from packet traces collected on our corporate network.

A. Flow collection

The prototype flow monitor is a simple device driver interposed in the network stack. This acts as an ETW (Event Tracing for Windows)[3] *producer*, posting events as packets enter, traverse, and leave the stack. An ETW *consumer* then runs in user-space and synthesises flow data using the packet data events from the provider. The ETW subsystem is quite efficient, imposing a CPU load of around 5% when posting 20,000 events per second[4]. Furthermore, by restricting kernel-space instrumentation to posting events based on packet headers, the definition of a flow resides solely in the user-space

[3] A low overhead event posting infrastructure [6].

[4] It is quite capable of posting an event for every context switch without crippling performance for example.

component where the flow data is synthesised. Decoupling the basic traffic monitoring facility from the flow data collection allows the definition of flow to be easily modified, potentially even on a per-query basis.

B. Route collection

Large enterprise IP networks typically manage routes within the network using a link-state routeing protocol as they are widely supported, well-understood, and relatively easy to manage. Link-state routeing protocols operate by having each router advertise its adjacent neighbours to all other routers, collect the corresponding adverts from the other routers, and then run a shortest path algorithm across the resulting database of links to determine routes to all destinations. As LSAs are flooded to all local routers[5], our prototype passive collector simply snoops the OSPF protocol, building and maintaining the current network topology.

C. Data aggregation

Flow and topology information are to be aggregated to construct a complete view of the traffic in the network, and made available to applications built over the platform. The aggregated dataset logically contains the traffic flow matrix $A_{ij} = \{\text{bandwidth from src } i \text{ to dst } j\}$, each entry annotated with the current route from source to destination. Note that the size of most enterprise networks makes such a dataset too large and dynamic to simply backhaul all the collected data to a single point.

To investigate the various trade-offs in this part of the system we have begun to simulate the distribution of flow and route data over a dynamic topology, and to investigate appropriate API support for applications built over the platform. Among the trade-offs to explore here are:

- The tension between efficient resource usage by the platform (probably best achieved by making use of the natural hierarchy in the network's design), and a self-organizing robust data store (as provided by a DHT for example).
- Which datasets should be distributed, and how much distribution is appropriate. For example, in a very stable network with changeable flow dynamics, it would probably be better to distribute the topology to all nodes rather than try to distribute flow data widely. Current data suggests that some middle way is probably sensible: centralize flow data for the local area at a machine in the local area, and distribute topology data to all such machines.
- How accurate the system can be made, given the problems of lack of coverage (it will be impossible to instrument all traffic sources in the network), lag (it takes time for changes in flow and especially topology to percolate through the system), and so on.

This simulator forms the basis of the demonstrator, described below.

[5]Non-local routers learn of each other through summarization mechanisms not detailed here; in the face of such mechanisms, it is necessary to monitor the routeing protocol from many network vantage points.

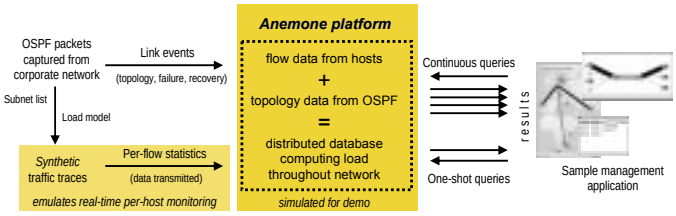

Fig. 2. Outline flowchart for demo. Pre-prepared traces (generated from live data) are fed through a simulation of the *Anemone* platform, which is interrogated by a simple sample management application.

V. Demonstration

We demonstrate a network management application running over a simulation of the *Anemone* platform, providing a variety of visualizations of a running network. Topological data is taken from traces of OSPF data collect by our prototype running in our corporate network; flow data is synthesised based on traces collected from both client and server machines in our network.

The initial visualization presented is an animated topology of the network, with line thickness proportional to link load. Aspects of the platform are demonstrated by enabling: (*i*) views of the *top-N* loaded links and communicating hosts; and (*ii*) *click-through* on links for a breakdown of the link's load in terms of the neighbours sourcing and sinking the link's load. The simulator also monitors and displays the inaccuracy in its calculated traffic matrix that results from instrumentation of only a fraction of the end-systems in the network. The accompanying poster summarizes the information in this proposal concerning the problem space, state of the art, and proposed solution, in addition to describing the design space and challenges of this approach.

References

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

[2] R. Huebsch, B. Chun, J. Hellerstein, B. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. Yumerefendi. The architecture of PIER: an Internet-scale query processor. In *CIDR*, Jan. 2005.

[3] 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.

[4] 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.

[5] C. Logg, L. Cottrell, and J. Navratil. Experiences in traceroute and available bandwidth change analysis. In *NetT '04: Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting*, pages 247–252. ACM Press, 2004.

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

[7] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Transactions on Networking (TON)*, 12(1):2–16, 2004.

[8] R. Teixeira and J. Rexford. A measurement framework for pin-pointing routing changes. In *NetT '04: Proceedings of the ACM SIGCOMM Workshop on Network Troubleshooting*, pages 313–318. ACM Press, 2004.

[9] 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.