NetAssay: Providing New Monitoring Primitives for Network Operators

Sean Donovan
Georgia Institute of Technology
sdonovan@cc.gatech.edu

Nick Feamster
Georgia Institute of Technology
feamster@cc.gatech.edu

ABSTRACT

Home and business network operators have limited network statistics available over which management decisions can be made. Similarly, there are few triggered behaviors, such as usage or bandwidths cap for individual users, that are available. By looking at sources of traffic, based on Domain Name System (DNS) cues for content of particular web addresses or source Autonomous System (AS) of the traffic, network operators could create new and interesting rules for their network. NetAssay is a Software-Defined Networking (SDN)-based, network-wide monitoring and reaction framework. By integrating information from Border Gateway Protocol (BGP) and the Domain Name System, NetAssay is able to integrate formerly disparate sources of control information, and use it to provide better monitoring, more useful triggered events, and security benefits for network operators.

Keywords

Network monitoring, network management

1. INTRODUCTION

Network management is a complex area. On stub networks, where the network is only a client to other networks, monitoring user behavior is limited. Bandwidth monitoring at the port or address level is among the more complex operations that are available. Similarly, triggered behaviors such as usage caps are limited. More complex behavior is available, primarily in highly specialized security appliances, however these are not general purpose devices.

What is needed is a generalized system that can monitor network activity and produce behaviors based on high level policies. For instance, a network operator would like to know how much video traffic each user is consuming in a month, and, if they reach a specified limit, limit their bandwidth for the rest of the month. Right now, there is no easy way of doing this.

SIGCOMM '14 Chicago, Illinois USA

NetAssay is being developed to make it easy for network operators to implement such policies. In the video bandwidth example, NetAssay needs to decide what network flows are video traffic. Traffic coming from `googlevideocontent.com` will be classified as video, as could any traffic associated with AS 2906, Netflix's AS.

More interesting triggered behaviors are also possible. For instance, a network operator has blacklists of ASes and domains that are associated with malicious behavior, but do host some legitimate content that should still be accessible. In this case, if a connection was destined for an IP associated with a potentially malicious AS, the network could automatically redirect traffic to a virus scanning appliance, rather than needing to send all traffic through the appliance. Another use would be to set a limit for amount of data that would be sent to a blacklisted AS before blocking connections, due to suspected bad behavior.

Similar triggered behaviors have been described in prior works [5][1], however this has been limited to network security security devices.

2. EXAMPLE BEHAVIORS

Before going into the design, a few examples follow. This is in Pyretic syntax [3], where `>>` is sequential composition (operations run in order) and `+` is parallel composition (operations run in parallel).

```
traffic_in_bytes(to = 'BobsPC',
                 content = 'video',
                 timespan = 'monthly') > 10**9 >>
    rate_limit(to = 'BobsPC',
               content = 'video',
               bandwidth = 65536)
```

The example above is a variation on the video bandwidth limitation example. In this case, we only care if Bob's PC's mac address is downloading more then 10 gigabytes of video traffic in a month. If it does reach this limit, then we want to limit the bandwidth to 64 kilobytes per second.

```
match_AS(pkt.srcip, AS = '12345')) >>
    drop()
```

The second example is simpler. We are see if a packet came from a AS 12345. If so, we want to drop it. Implicitly, if we do not drop a packet, it will be forwarded.

3. DESIGN

NetAssay is SDN-based and works on top of the SDX platform [2]. NetAssay is divided into three parts, as can be seen

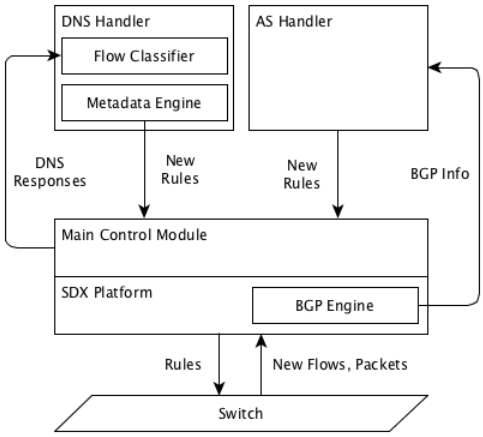

Figure 1: NetAssay architecture

in Figure 1. There are two monitoring modules, described below, that provide traffic identification and monitoring information to the main control module (MCM). The MCM is responsible for combining the results from the monitoring modules, and implementing the reactive policies defined by the network operator.

In the examples above, the MCM would be responsible for integrating the resulting policies low level policies from the DNS handler and AS handler for the `traffic_in_bytes()` and `match_AS()` and sending the matched flows or packets to the the `rate_limit()` and `drop()` actions.

3.1 DNS Handler

The DNS handler contains two parts, a flow classifier based on part of FlowQoS [4] and a metadata engine. At startup, the DNS handler creates a rule in all ingress network devices to make a copy of DNS responses, forward one to the actual destination, and forward the second to the DNS handler, and to the flow classifier in particular. The flow classifier parses the DNS packet, extract the domains and associated IPs, and classifies traffic from the IPs based on the domain name. For instance, if during the first example in section 2 a DNS response from `googlevideocontent.com` with associated IP of 198.51.100.8 was seen, traffic coming from 198.51.100.8 would be assumed to be video until the DNS response time-to-live (TTL) was reached.

The metadata engine handles policy implementation for DNS-based classification. The metadata engine is responsible for breaking down the high-level policies passed form the MCM into the constituent components, and handling any changes that my occur dynamically (*i.e.*, when a new DNS response comes in). It is also responsible for aggregating data, such as aggregating the number of bytes transferred by video sources.

In the first example in section 2, the new DNS response from `googlevideocontent.com` would cause the metadata engine to create a new byte counter for flows between 198.51.100.8 and Bob's PC. The new counter would be added to the previous policy and added to any prior video byte counter aggregation. The total would be passed up the the MCM to make any decisions based on the total amount of video traffic to Bob's PC.

3.2 AS Handler

As NetAssay is based on the SDX platform which contains a BGP-engine, we have access to BGP routes. The primary reponsibilities are similar to those of the DNS handler, in that the AS handler needs to break down the high level policy into its consistuent components and returning them to the MCM.

In the second example from section 2, the AS handler would be responsible for breaking down the `match_AS()` into the constituent components. These components would be a series `match()` actions on source IP addresses from packets to the IP ranges that AS 12345 was advertising. The parallel composition of all of these `match()` would be composed with the `drop()` action by the MCM.

Many networks, in particular home or small business networks, BGP information is not accessible. As such, this handler can be seen as optional.

4. PROGRESS AND FUTURE WORK

Work on NetAssay is ongoing. Design work is progressing. The DNS flow classifier is complete, though the classification data set is small and video-focused at the moment.

There are some inherent limitations, but will be resolved thanks to progress on other project. In particular, SDX is limited to a single switch right now, however this may be a minor issue due to ongoing SDX development.

Future work, beyond finishing currently planned work, is to proactively perform DNS look ups from various public DNS sources in order to get a broader data set. While conflicts between the handlers policies are unlikely, handling conflicts needs to be addressed in the future. Along the same lines, in cases where bandwidth is being counted, preventing 'double counting' between the two handlers is also necessary.

5. REFERENCES

[1] J. R. Ballard, I. Rae, and A. Akella. Extensible and scalable network monitoring using opensafe. *Proceedings of USENIX Internet Network Management Workshop/Workshop on Research on Enterprise Networking*, 2010.

[2] A. Gupta, L. Vanbever, M. Shahbaz, S. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett. Sdx: A software defined internet exchange. *Proceedings of the ACM SIGCOMM 2014 conference*, 2014. To Appear.

[3] J. Reich, C. Monsanto, N. Foster, J. Rexford, and D. Walker. Modular sdn programming with pyretic. *USENIX ;login,*, 38(5):128–134.

[4] M. S. Seddiki, M. Shahbaz, S. Donovan, S. Grover, M. Park, N. Feamster, and Y.-Q. Song. Flowqos: Qos for the rest of us. *HotSDN 2014*, 2014. To Appear.

[5] S. Shin and G. Gu. Cloudwatcher: Network security monitoring using openflow in dynamic cloud networks. *20th IEEE International Conference on Network Protocols*, 2012.