NetQuery: A Universal Channel for Reasoning about Network Properties

Alan Shieh†, Oliver Kennedy†, Emin Gün Sirer

{ashieh,okennedy,egs}@cs.cornell.edu
Dept. of Computer Science, Cornell University

Although the configuration of modern networks has a significant impact on the performance, robustness, and security of applications, networks lack support for reporting these differences. For instance, ISPs that have made investments in high-speed routers, excess network capacity, fault-tolerant topologies, internal caches and middleboxes for DDoS protection are difficult to tell apart, at the IP level, from shoestring operations. The IP interface makes it difficult for ISPs to differentiate their service on any front except price, and forces customers to rely on ad hoc techniques to discover the properties of their network.

Likewise, configuration differences between end hosts are of interest to network operators: hosts configured without firewalls and virus checkers can launch internal attacks on a protected network, while hosts with promiscuous interfaces and untrusted network stacks can monitor network traffic. Network operators lack the communication channel necessary to establish the capabilities of their clients, and need to rely on brittle and insecure techniques, such as connection hijacking and interstitial pages, to extract information from their users.

We are developing NetQuery, a system that provides a channel for distributing and reasoning about the properties of network participants. NetQuery is a distributed tuplestore for collecting, querying and disseminating such properties (Figure 1). The tuplestore captures the properties of any network device used to source, sink, or forward packets, including access points, switches, routers and end hosts, as a set of attribute-value pairs. Clients can query the attributes of the network elements to establish whether their requirements of the network are satisfied, as well as establish triggers to detect future modifications to network state. NetQuery enables any third-party to tag any network device with custom attributes, and allows different network information services to coexist on the Internet without the need for a central administrator to mediate contradictory statements. The language used to express properties is extensible, enabling NetQuery to support new network elements and properties as networks evolve.

Reasoning about properties of remote network elements only makes sense when there is an established basis for trusting their claims. NetQuery provides a flexible framework for establishing the verity of such claims. Though this framework also supports reasoning

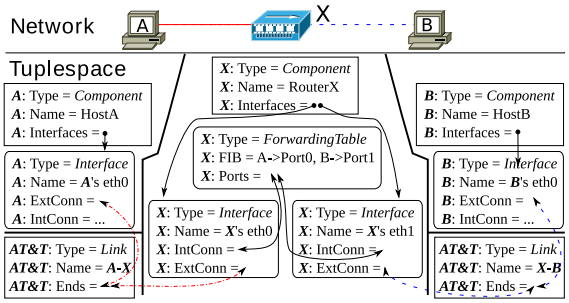

Figure 1: **A simple network and its tuple-based representation.** Every network element has a corresponding tuple that describe its current configuration and state.

over legacy networks, we focus on a "clean-slate" design where all networked components are equipped with inexpensive secure hardware coprocessors, such as the $< \$10$ Trusted Platform Module (TPM). This trusted hardware serves as a root of trust for generating verifiable claims about a remote machine; NetQuery translates these low-level assurances into strong guarantees over the network.

Three sample scenarios illustrate the types of applications that NetQuery enables:

- *Path properties.* Deep packet inspection enables ISPs to datamine and manipulate network flows. A privacy conscious user can use NetQuery to determine how a web session is monitored.
- *Routing policies.* ASes expect their neighbors to conform to contractual agreements such as peering and mutual backup. ASes can use NetQuery to detect or prevent violations.
- *Topology changes.* Events such as link failures or wireless roaming can affect a network's capacity and capabilities. NetQuery can notify endpoints of these changes so that they can adapt accordingly.

We have built a NetQuery prototype, consisting of a tuplestore server; routers, switches, and end hosts that exchange configuration information with the tuplestore; and applications that use this information to reason about the network. The tuplestore efficiently disseminates network properties: a small number of tuplestore servers can service the routing changes of all devices within a POP. Our applications demonstrate that NetQuery provides an expressive programming model for checking the properties of a network.

†Student author