On Information Hiding and Network Management*

Kenneth L. Calvert and James Griffioen
Laboratory for Advanced Networking
University of Kentucky
{calvert,griff}@netlab.uky.edu

Keywords

network management, scalability, privacy, network architecture.

ABSTRACT

No single administration controls the entire Internet. Instead, *competing* providers work together to enforce of a wide variety of network management policies, including policies that limit the flow of management information itself. In many cases these policies are designed to keep information about the state of the network from "leaking" outside the network. In this position paper, we consider the ramifications of such information-hiding policies for network management. We discuss mechanisms that might be used to enforce such policies, and argue for an open access policy.

1. INTRODUCTION

For the purposes of this paper, *network management* is about keeping the network in a desired state. This problem has two separable aspects: (i) *gathering information* about the state of the network; and (ii) using that information to *control* the configuration of the individual elements (routers, switches, hosts etc.) that make up the network. The Simple Network Management Protocol (SNMP) [5] supports both aspects: the SNMP SET message provides control, while other parts of the protocol all support information-gathering. In this paper we focus on the former, i.e. aspect (i). We observe in particular that access to information about the network state is useful not only for network management, but also for other purposes such as enhancing application performance end-to-end.

Presently no single administration controls the entire Internet; instead, each service provider manages that part of the network it controls, which we refer to as its *domain*. Paradoxically, service providers must cooperate to provide the basic end-to-end service of the Internet, but at the same time they are competitors. As such, each has little incentive to help others gather information about the state of its domain; in fact, there are well-known incentives to *hide* such information.

In this position paper we explore this "tussle" between providers, who want to control or even prevent access to their network management information, and others who might make use of such information. Our **position** is that allowing access to such information provides significant benefits for management of the global Internet as well as for other purposes, but that traditional access control mechanisms are not a good solution for the problem of hiding sensitive information. We therefore propose an approach that grants access to those outside the domain, but allows providers some control over what information is revealed. The approach focuses on *what* information is made available, and how, rather than on *who* is allowed to access it.

The rest of this position paper is organized as follows. In the next section we describe a simple model of the network management environment and briefly discuss motivations and mechanisms for hiding information. In Section 3 we consider the other side of the tussle, i.e. the benefits of making management-type information available. Section 4 describes an approach that makes management-type data available to all packets on or near the fast path; hiding is implemented by applying transformations at domain boundaries. We discuss some possible transformations and how they might be implemented using Ephemeral State Processing [2] as an example. Finally, Section 6 concludes the paper.

2. MODELS AND ASSUMPTIONS

The discussion that follows is shaped by the existing Internet architecture, but our intent is that it should apply in a more general context, such as the next generation network architecture.

We assume a model that distinguishes between the part of the network being managed, which we refer to as the *domain*, and the rest of the world. Each node in

*Work sponsored in part by the National Science Foundation (EIA-0101242 and CNS-0435272), the Kentucky Science and Engineering Foundation, Intel Corporation, and Cisco Systems.

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

the network (router or end system) either belongs to the domain or does not; links that connect a node belonging to the domain to one or more that does not are border links. A router that has is connected to one or more border links is called a *border router*. The "perimeter" of the domain is defined by its border links. We assume that each border router knows which of its interfaces are part of the perimeter.

We assume that each node in the domain maintains a local store of information for management purposes (e.g. MIB variables). Some of this information may be associated with sub-components of the node (network interfaces) while some is associated with the node as a whole. By "information hiding" here we refer to restrictions on the ability of entities outside the perimeter of the domain to *read* this information. (We do not consider writing, i.e. configuration of nodes, in this paper.)

It is an observable fact that network operators are sensitive about revealing data about their network and its operation. Traffic traces, network topologies, and the location and nature of network interconnections are all considered confidential, and must be "sanitized" before being made publicly available (e.g. to researchers), if they are revealed at all [11, 13]. There are several reasons for this:

- **Competitive pressure**: Most providers are in business to make money; as such, they typically try to reveal as little as possible to their competition about the way they operate and their relationships with customers. For example, comparative information about the internal performance of different networks might be exploited for marketing purposes by the provider with better numbers.
- **Customer Privacy**: Providers worry about revealing information about customers' communications that might be considered sensitive. They do not want outsiders to be able to infer traffic patterns, for example, or even which customers are talking to which.
- **Safety and Security**: Information about which version of software runs on a network element might help an attacker determine a way to compromise the element. Such hints can be gleaned from surprisingly innocent-seeming data—for example, the contents of ICMP meessages.

The standard approach to restricting access to information is to distinguish between those parties who are allowed to obtain the information and those who are not. This requires that a mechanism be put in place to mediate all access to the information. This mechanism (i) determines who is responsible for each request to access restricted information (authentication); (ii) determines whether the requester is allowed access according to the relevant *policy* (authorization); and (iii) either allows or denies the requested access to the information.

In the present architecture, SNMP plays this role. SNMP provides for three methods of authentication: (1) via packet labeling, in which a requesting packet is considered "authentic" (i.e. sent by an authorized party) if it carries a recognized "community" string; (2) via source IP address, in which a requesting packet is considered authentic if its source address is on a list of "trusted" hosts; and (3) via cryptography, where a packet is considered authentic if it carries a token that can only be generated with knowledge of a secret shared with the access control mechanism. Obviously these mechanisms vary in terms of the level of security they provide. Cryptography offers the greatest security, but also brings with it additional costs in two forms. One is *computational cost*: verifying public-key signatures requires significant computing resources, and gives rise to the possibility of a denial-of-service attack by bombarding a network element with bogus requests. Such an attack can prevent legitimate work (e.g. management requests) from being processed. Other cryptographic approaches may consume fewer resources, but require the distribution of secret keys to participating nodes. This represents a different cost, less often considered but no less important, namely *administrative overhead*: someone has to specify, configure, and maintain the policies and related information (e.g. secret keys) on network elements and management stations. All three SNMP mechanisms have this cost.

3. BENEFITS OF NON-HIDING

Clearly, exposing information to nodes *inside* the perimeter is necessary for network management. In particular, network management stations within the domain require access to local network statistics in order to detect failures, diagnose and isolate configuration errors, track link and server loads over time, identify intrusions and attacks, monitor service levels, etc. Although local network managers stand to gain the most by exposing information within a domain, we note that distributed services/applications within the domain may also utilize this information to optimize performance (e.g., load balancing requests across a server farm).

One might conclude that it is best to hide all management statistics from nodes *outside* the domain perimeter, while permitting all access from inside. However, we argue that this policy is more restrictive than necessary. In many situations, *some* information can be "leaked" without compromising privacy, safety/security, or competitive advantage. Moreover, leaking non-sensitive information (i.e., not private information of customers) to nodes outside of a domain can benefit the system as a whole, allowing it to operate more efficiently or offer services not possible with a "black box" network. In that sense, it is in the best interest of a domain to be a good network citizen, contributing information for the greater good of all domains and benefitting from the information exposed by other domains.

We have found that high-level, abstracted information can be as useful as more detailed, fine-grained information. Indeed, it is well known that information hiding can *improve scalability* [3]. In many cases the requesting entity is only interested in a summary of the information collected from a set of nodes, or is only interested in the answer to a very specific question. In such cases exporting the information from all nodes requires more bandwidth and actually hurts scalability. For example, a sender transmitting simultaneously to a large group may want to know the range of throughputs observed

across all receivers. In this case, the sender does not need to see all values; rather, a small set of "distilled" values (min, max, mean) suffices. As another example, an application might seek the available capacity of the bottleneck link along a path to a destination. Information about the identity of the nodes on either end of the link is extraneous, and nothing is lost by hiding it.

Finally, consider the problem of determining whether two network paths intersect and the queue length at the intersection (if any). Again, we are only interested in the existence of an intersection point, and the ability to query it for its queue length; this does not require learning the identity or location of the intersection point. Instead, the intersection, if it exists, can identify itself with a unique id—something like a cookie issued by a web server—which, when seen later in a request for the queue length, will trigger a valid response.

Benefits that may accrue from sharing information of this kind outside the domain include faster diagnosis of global problems, improved/optimized application performance, enhanced traffic engineering capabilities (e.g., routing/load balancing), and reduced loads at monitoring stations (because the monitored information is preprocessed, filtered, summarized, etc. before arriving at the monitoring station).

Consider the well-known hot-potato routing strategy, in which a domain hands off packets to a neighboring domain as quickly as possible based on its own internal network costs, without knowledge or consideration of the costs associated with the resulting path through the neighboring domain. Given knowledge of the approximate costs between ingress and egress points of the neighboring domain, a more intelligent selection can be made, possibly resulting in improved end-to-end performance for the customer.

As an example of improved application performance, the canonical example is congestion control. The Internet was not designed to provided explicit feedback about congestion (e.g., where it is occuring or the level of congestion). Given the importance of this problem, a variety of (congestion control specific) network-level approaches have been proposed [8, 6]—some providing explicit feedback, others providing intentional, but implicit, feedback. Through explicit but anonymized information about a path's congestion level—e.g., giving link loss rate and the number of flows sharing the link, while hiding the associated link IDS (or at most revealing codenames/aliases of the associated link)—end systems can react quickly and precisely to congestion anywhere in the system without discovering the location responsible for the congestion.

4. SCALABLE INFORMATION ACCESS

So far we have argued that traditional access control methods, where each node individually authenticates requests for information and implements a local policy, fail to scale both in bandwidth and in administrative costs. We have also observed that in many cases detailed information, which a provider might consider sensitive, can be rendered innocuous by transformations that preserve and even enhance its utility. Elsewhere [2] we have argued that summary transformations of this kind are useful for network management, and partially address a growing need—recently noted by the Internet Architecture Board—for "scalable techniques for data aggregation and event correlation of network status data originating from numerous locations in the network" [1]. In addition to network management, the ability to extract such information from a domain would be useful in a variety of ways to applications and other domains.

These considerations lead us to suggest a model with the following characteristics:

- Any packet can potentially request information. This implies that packet processing related to information retrieval must be handled on, or very near, the fast path, in order to preclude denial-of-service attacks. In other words, the mechanism that provides access to network management information must be extremely lightweight.
- Information can be processed, as well as read, on the fly as packets pass through the network. This enables the distributed computation of summary information to improve scalability as well as implement hiding.
- All retrieved information is (potentially) transformed as it crosses the domain perimeter. This allows the provider to control what is exposed.

Obviously this model is quite different from the existing SNMP-style model in which a single management station queries nodes one at a time to gather information. Where the traditional model focuses on *who* is allowed to access *what* information, and either allows or denies access altogether, our model allows universal access, and relies on various *transformations* to implement the information-hiding that providers require.

In this section we present three general information-hiding techniques that can be used individually or in combination by a domain to hide certain information while exposing allowable information. In the next section we shall describe a way of implementing these techniques using a lightweight router-based primitive.

4.1 Aggregation

Aggregation involves hiding individual values collected from a set of nodes, and returning instead a single value that is the result of applying a request-specific function (e.g., minimum, maximum, average, sum, etc.) to the individual values. Although the function is applied to internal domain values, the result that is returned is a combination or aggregate of the original information.

Example uses of aggregation include monitoring (or probing) the network to answer questions like "what is the maximum loss rate among all multicast receivers?" "how many receivers are in the multicast group?", "how many nodes are experiencing congestion?", or "what is the average server load?"

Note that aggregation serves multiple purposes. First it hides the values stored at each of the contributing nodes. In fact, it hides the number of nodes contributing values to the computation. Clearly aggregation protects hidden information from being exposed, but it has the added benefit of simplifying the interface by which the

requester learns information about a (potentially) large group. Second, it allows the required processing to be distributed across nodes in the network, thereby offloading work from the requester (much as multicast offloads the task of packet duplication).

4.2 Anonymization

Anonymization refers to obscuring the identities of individual nodes, while maintaining the ability to differentiate among nodes. For example, a link along a path might admit performance problems—a high loss rate—by returning its loss rate and a "cookie" that is meaningful to the lossy link, but meaningless elsewhere. The cookie might be included in subsequent probes of alternative paths to determine whether the bad link is part of the alternate path. The scope of such a cookie would be the domain in which it was generated; nodes inside the domain would know the mapping (the particular transformation could be encoded in the cookie itself) between nodes or addresses and cookies. At a minimum, the anonymized value returned must be the same for all packets that request the value over some (potentially small) period of time. This requirement is necessary to match up results from multiple packets. Beyond this requirement nodes are free to reveal as little or as much as they like. For example, a node may reveal that it is at the University of Kentucky (in its IP range), but not reveal which machine it is or which subnet it is on.

The normal case we envision would be for anonymization to occur at border routers. However, if certain values are not to be exposed within the domain—at least not through the lightweight network management mechanism—then the anonymized version of the value would be returned by each node, even within the domain. Of course, the non-anonymized version could still be accessed through heavyweight approaches such as SNMP.

4.3 Abstraction

Abstraction is useful when a set of nodes or links are not only to be hidden, but should be treated/viewed as a single (virtual) entity. How the virtual entity is "constructed" from the actual component parts that it represents is domain-dependent and must be defined by the domain administrator.

Instead of revealing information about all the links on the path from one of the domain's ingress points to one of its egress points, the domain administrator may want to represent the path as a single (virtual) link whose characteristics are derived from those of the individual links. For example, the end-to-end delay of the virtual link may be computed by summing the end-to-end delays of the individual links. Similarly, the loss rate of the virtual link could be the maximum of the rate of all links on the path.

It should be noted that aggregation differs from abstraction in the sense that aggregation allows an outside node to define a function to be applied to the actual values within a domain. In the above example of the maximum loss rate along a path, the same result could be obtained by using aggregation. On the other hand, abstraction takes a set of nodes and/or links and converts them into a virtual entity by applying a transformation that is specific to the domain and to the type of information requested.

In particular, with aggregation, ESP packets carry only summary information. Abstraction, on the other hand, requires that the detailed information in a packet be *modified before leaving the domain*. Thus, border routers are key to the abstraction mechanism and represent the point at which abstraction policies (defined by the local domain administrator) are enforced. In particular, border routers are responsible for converting information requests into an internal request that will generate the information needed to represent the abstract object (node or link). We discuss this in more detail in the next section.

5. IMPLEMENTING HIDING

We argue that the mechanisms needed to support the three approaches presented above should be built into routers, and in particular, should be implemented on port cards (as opposed to in a centralized processor). Several router-based data collection/aggregation protocols have been proposed in the context of multicast and data collection [3, 4, 10, 7, 12]. Although these may work well for aggregation, they are not necessarily capable of supporting the other information hiding approaches we propose. Moreover, they are not generally designed to process packets at line speeds and thus may be susceptible to Denial-of-Service attacks. In this section we describe a mechanism that is capable of supporting all three information hiding techniques and is designed to be impervious to DoS attacks (i.e., can be executed on every packet at line-speeds).

5.1 ESP-NM

In earlier work, we proposed Ephemeral State Processing (ESP) [4] as a lightweight and scalable way for end systems to perform distributed processing inside the network. ESP provides the ability for packets to store and compute with small, fixed-size chunks of short-lived information on router interface cards. The information is kept in an associative store of (tag, value) pairs referenced by the tag field. The basic idea is as follows: as packets travel through the network, they create, modify, or retrieve these chunks at interfaces of routers along the path from source to destination. This information, once created, exists only for a short time (say 10 seconds) and can only be used by subsequent packets within that time interval. Each packet carries a single, short, instruction code that indicates the operation to be performed on the stored state and/or the values carried in the packet. Example instructions include the *count* instruction, which increments the value stored with a tag carried in the packet; and the *find* instruction, which can be used to find the maximum or minimum value bound to a tag carried in the packet at the nodes along the path, as well as the address of the node (interface) at which it was stored. Approximately a dozen different instructions have been defined to date and the instruction set continues to evolve.

All ESP packets are either forwarded straight through to their destination or are silently discarded based on the outcome of their processing. The computation initi-

ated by any ESP packet is simple and requires strictly bounded resources (including the space-time product of state storage); this enables it to be implemented on or near the fast path. ESP processing never modifies any field of the network header of the packet. The power of the service comes from the ability to sequence these computations in time and space, by sending packets containing different instruction codes.

In follow-on work [2], we proposed enhancements to the basic ESP service that enabled its use for network monitoring and management; we call the resulting service ESP-NM. The primary enhancement enables access to the network management information base (MIB) at each node in the network. That is, in addition to providing access to ephemeral state values at a node, ESP instructions can reference MIB variables at a node. (We restrict MIB information accessed through ESP to be read-only; to maintain efficient processing, any centrally-defined values would be duplicated at each line card and updated upon every change.)

To understand how the service might be used, consider the problem of determining whether two paths go through any common router interfaces (i.e., intersect). Initially an ESP packet is sent along the first path depositing a value in the incoming and outgoing interfaces of every router. Shortly thereafter (within the state lifetime), an ESP *find* packet is sent along the second path looking for the maximum value bound to the tag used by the first packet. If the paths intersect, the second packet will arrive at its destination containing the value and the address of the interface at which they intersect. If there is no intersection, the packet will arrive at its destination containing no value and no address.

5.2 Implementing Transformations

The following briefly describes how ESP-NM could be used both as the query protocol and to implement all three information hiding techniques.

Aggregation. As described in [2], a two-phase computation can be used to apply any associative and commutative operation to the values stored at a set of nodes. In the first phase, the nodes all send *count* instructions toward the collecting node. This sets up a virtual tree structure by establishing "child counts" at each intermediate node. In the second phase, each node transmits a *collect* instruction carrying its value toward the receiver; at each intermediate node, each incoming *collect* packet's value is added to the sum and the child count is decremented; when a node's child count reaches zero, its sum value is written into the *collect* packet, which is forwarded on toward the receiver. When the receiver's child count reaches zero, its sum contains the total for the group.

Anonymization. Implementing anonymization with ESP requires that certain requested information carried in ESP packets be rewritten by border routers according to a well-defined transformation. For example, an ESP packet carrying the identifier (IP address) of the bottleneck router along a path would be rewritten at the edge of the domain to return a transformed version of the router's identifier. Similarly, incoming packets that carry anonymized IDs have to be rewritten at the border so they have the actual address. In this case, border routers must know the mapping between the non-anonymized value and the anonymized value. To avoid analysis of the mapping by outsiders, the ID can be time-dependent and continually changing since it only needs to be valid long enough for subseuquent ESP packet (transmitted soon after the first one) to find the same anonymized ID and obtain the desired information.

Abstraction. Like anonymization, abstraction involves modifying ESP at border routers. In this case, however, the border router transforms an instruction that might reveal sensitive information about a domain into one that casts the information in a less sensitive form. One possibility is to use an ESP *instruction stack*, where ESP instructions are pushed on entry to a domain and popped on exit. The basic idea is that each ESP packet is checked at the input interface of a border router, and if it would reveal sensitive information, the router pushes a new ESP-instruction onto the ESP instruction stack; the new instruction is designed to compute the desired value, but over a virtual entity. When the packet arrives at a border router on the way out of the domain, the top instruction is "popped" along with its result, which is then used in processing the original instruction, and the packet is forwarded normally.

As an example, consider an ESP instruction designed to find the loss rate of the lossiest link along a path. At each hop, the number of packets dropped is divided by the number of packets forwarded and the result compared to the current value in the packet; if it is greater, it is written into the packet. If a domain administration does not wish to reveal loss rates for individual links, at the border it can push a new instruction that sums up the number of dropped packets and the number of transmitted packets on each link. At the egress router, the instruction is popped, the sums are divided and the resulting value is used as the loss rate for a virtual link that corresponds to the entire domain.

Application of this transformation may be triggered by lookup of incoming ESP instructions in an abstraction policy table. The pushing and popping conversion at the border can be done quickly and efficiently (similar in many ways to tag stacks in MPLS [9]).

In the case of anonymization and abstraction, border routers need to process ESP packets that cross the domain perimeter differently from those that do not. In the next subsection we describe how this capability can be implemented.

5.3 Perimeter-Crossing Packets

ESP processing may be handled differently on a packet depending upon whether it is crossing a domain perimeter or not. In particular, in case of abstraction a new

instruction must be pushed on ingress and popped on egress from the domain. Since a router may have multiple border links as well as multiple "inside" links, some means is required to determine, prior to the ESP processing on the input interface, whether the packet will traverse the domain perimeter. One possibility would be to examine the destination address and recognize those addresses inside the domain. That approach, however, requires that a list of internal addresses be maintained and compared to each ESP packet.

A better method is to compare the links on which the packet is received and transmitted; if one is a border link and the other is not, the packet is crossing the perimeter of the domain. Of course, the outgoing link is not known until after the forwarding lookup is completed. This implies that ESP processing in incoming packets should occur after the forwarding lookup. More precisely, for input ports, processing should occur as follows:

```
(packet arrives on link l)
    validate packet;
    lookup destination to obtain outgoing link e;
    if (l is border and e is not)
        ⟨ingress ESP processing⟩;
    else if (e is border and l is not)
        ⟨egress ESP processing⟩;
    else
        ⟨normal ESP processing⟩;
```

Here "ingress ESP processing" means un-anonymizing values and pushing an instruction for a virtualizing computation on the ESP instruction stack as necessary. Similarly "egress ESP processing" includes anonymizing sensitive values like node identifiers, and popping the ESP instruction stack and completing the virtualizing computation, if any.

6. CONCLUSION

In prior work we have dealt with network scalability mechanisms that support access to group information in a distilled or abstracted form [3, 2]. We have observed that information commonly related to network management can be useful outside its domain of origin; however, providers have incentives to restrict access to such information. We argue that traditional heavyweight access control mechanisms, which might allay concerns about revealing sensitive information, are inappropriate due to high costs. Based on our experience with other scalability mechanisms, we prefer an approach featuring universal access and hiding sensitive information through network-based transformations. We have sketched a few brief examples of such transformations, and mechanisms through which they might be implemented. Our ongoing work considers the details of how such mechanisms might be deployed in existing and future networks.

7. REFERENCES

[1] R. Atkinson and S. Floyd. IAB Concerns and Recommendations Regarding Internet Research and Evolution, August 2004. RFC 3869.

[2] K. Calvert and J. Griffioen. Scalable network management using lightweight programmable network services. *Journal of Network and Systems Management*, 14(1), March 2006. (Special issue on management of active and programmable networks.).

[3] K. Calvert, J. Griffioen, B. Mullins, A. Sehgal, and S. Wen. Concast: Design and implementation of an active network service. *IEEE Journal on Selected Areas of Communications*, 19(3):426–437, March 2001.

[4] K. Calvert, J. Griffioen, and S. Wen. Lightweight network support for scalable end-to-end services. In *ACM SIGCOMM 2002*, pages 265–278, August 2002.

[5] J.D. Case, M. Fedor, M.L. Schoffstall, and C. Davin. Simple Network Management Protocol (snmp), May 1990. RFC 1157.

[6] D. Katabi, M. Handley, and C. Rhors. Congestion control for high bandwidth-delay product networks. In *SIGCOMM 2002*, pages 89–102, August 2002. Pittsburgh, USA.

[7] John C. Lin and Sanjoy Paul. RMTP: a reliable multicast transport protocol. In *Proceedings of IEEE Infocom*, March 1996.

[8] K. K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ecn) to tcp, September 2001. RFC 3168.

[9] E. Rosen et al. MPLS label stack encoding, 2001.

[10] T. Speakman et al. PGM Reliable Transport Protocol specification, December 2001. RFC 3208.

[11] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with with Rocketfuel. In *SIGCOMM 2002*, pages 133–145, August 2002. Pittsburgh, USA.

[12] R. van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2):164–206, May 2003.

[13] J. Xu, J. Fan, M. Ammar, and S. Moon. Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme. In *2002 International Conference on Network Protocols (ICNP '02)*, pages 280–289, November 2002.