# Sapheniea: Simplifying Configuration Using Classes

*Cheng Tien Ee*
*Scott Shenker*
*Lakshminarayanan Subramanian*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-135
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-135.html

October 19, 2006

# Sapheniea: Simplifying Configuration Using Classes

Cheng Tien Ee
UC Berkeley

Scott Shenker
ICSI, UC Berkeley

Lakshminarayanan Subramanian
New York University

*Abstract*— **This paper describes the design of Sapheniea,[1] a framework that enables network administrators to easily implement policies in large-scale networks. The goal of Sapheniea is to capture as much configuration information as possible into a single parameter, which we define as *class*. The key idea is to categorize network traffic into different classes and embed the same class parameter as a configuration knob in routing. Network administrators only need to define the various classes, specify the relationship between them, and assign classes to links. In this paper, we provide two applications that illustrate how Sapheniea can be used in enterprise networks to perform: (a) access control within a domain; (b) traffic channeling through choke-points.**

## I. Introduction

Configuring large-scale enterprise networks is often a nightmare for network administrators. The role of configuration has increased substantially with the threats posed by worms, malware and attackers. To mitigate these threats, network administrators have been forced to deploy a range of middle-boxes including firewalls [3], NATs [14], VLANs [6] and NIDS [10], [13] that act on the network data path to detect and drop undesirable traffic. However, the current model of implementing access control in such networks is both cumbersome as well as error-prone; the process involves a complex combination of bridging, routing and manual installation of packet filters in routers along the data path [16], [17].

One of the fundamental reasons behind the difficulty in configuring such networks is the *incongruence* in configurability of the control and the data planes. Today, most of the middle-box configuration occurs in the data plane; operators install, along the data path, packet filters which need to be modified with each change in topology or routing. This incongruence stems from the fact that routing and filtering are treated as separate problems. The management burden increases with network size: in large-scale networks, low-level views available to operators, that is, filters at each individual router or switch, makes it difficult to reason about connectivity issues within the overall network, increasing the likelihood of misconfiguration errors.

In this paper, we describe the design of Sapheniea, a system that introduces an explicit configuration binding between the control and data planes using a single *class* parameter. In Sapheniea, traffic is first classified based on some policy, where the granularity of a class is configurable. Subsequently, middle-boxes in the data plane can apply policies tailored for each class, and the control plane allows route determination on a per-class basis. This explicit binding between the two planes eliminates the need for operator intervention when routes change.

Another important aspect of Sapheniea is the flexibility in defining the most basic *relationship between classes*, which is whether one class of traffic can be carried on links of another. This relationship can be arbitrary and captured by a *class-graph* which we require to be directed acyclic.[2]

### A. Motivating Examples

We motivate routing with classes using two examples: (a) access control, and (b) traffic channeling through choke-points.

**Access Control:** Operators add filters at routers to restrict access to certain resources in the network. The placement of filters is very much topology dependent, and legacy filters need to be accounted for when topology changes. The difficulty of grasping the complete picture is compounded with the fact that operators who installed seemingly cryptic legacy filters may no longer be available to assist in the process. Sapheniea can help resolve this issue, beginning with the assignment of classes to links and traffic. The level of indirection brought about by this assignment reduces the complexity of filters: rather than say "filter packets from 1.2.3.0/24, 1.2.4.0/24", we instead enlist the assistance of the routing protocol to say "never forward packets with classes lower than B". The association of a class to each link enables routing based on classes and provides the following advantages: (1) if, at a point in the network, no route to destination exists for a certain class, that information is automatically propagated by the routing protocol without the need to add filters at that location; (2) the lack of any route and hence visibility implies that traffic will not even begin to be forwarded through the network and subsequently dropped only at the end-host, reducing the impact on eligible traffic being carried; (3) the effect of link addition or removal on the admission of traffic is updated automatically by the routing protocol without further manual configuration; (4) by using the network and class-graphs, the operator can, at a glance, deduce and control the type of traffic allowed at a particular region.

**Traffic Channeling:** The traditional method used for channeling of packets through choke-points (such as firewalls, NIDS boxes) is the manipulation of physical network connections. This requires careful planning and execution at the ground level, and can typically result in tedious re-wiring of

[1]Sapheniea in Greek means "clarity".

[2]A cycle in the class-graph introduces a policy conflict across classes and is hence not allowed.

cables when the network topology changes. Worse, unintentional connections, such as those via wireless links, can bridge two otherwise separate networks [15]. Rather than attempting to control the physical connectivity, we instead only mandate that the network be well connected physically, then create logical networks with links that are a subset of the physical network. This is similar to VLANs, except that this concept is extended to layer-3 networks, and, as we show in §II the usage of classes and the ability to define inter-class relationships adds a degree of flexibility which will also be beneficial to VLANs. Upon entering the network, traffic can be classified based on some policy, with the traffic class subsequently resulting in a particular logical graph being used for forwarding. Thus, middle-boxes' placement becomes much less dependent on the physical topology: links' classes can be configured remotely such that traffic of interest can be channeled accordingly.

## II. The Sapheniea Framework

The Sapheniea framework consists primarily of three components: classes, rules and translation boxes. We briefly describe them below, then elaborate further in the subsequent subsections.

### A. Framework Components

**Classes** describe traffic to be carried as well as routes. The semantics of a particular class is defined by the network administrator, and relationships between classes are defined using a directed acyclic graph which we call the *class-graph*. The class associated with the route for a particular destination can alter as they propagate through the network, depending on, for instance, the links traversed or on some network requirement. Packets' classes are inserted at the first hop router, and removed just before leaving the network, thus no changes need to be made to end-hosts.

**Rules** govern the selection of routes at each router. With the introduction of class-graphs, routing effectively becomes multi-path, and various methods of route selection exist. These rules set up routing state within the network, much like today's OSPF [8] and RIP [7] and IS-IS [9].

**Transformation boxes** alter classes associated with routes as they are propagated through the network. These boxes are necessary when the network extends across perimeters, which can, say, encompass a region in an enterprise network where only trusted traffic can traverse. Transformation boxes can decide to trust and upgrade traffic after verification. Also, the semantics of a particular class may be different across perimeters, and thus mapping of one set of classes to another is necessary.

We elaborate on these three components in the following subsections.

*1) Classes:* The semantics of a class is determined by the network administrator. In traditional shortest-path routing, assuming that each link is assigned a weight of one, the class of a route advertisement received at a router is simply a

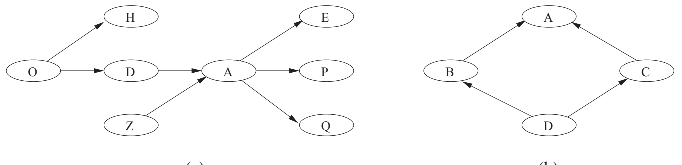

Fig. 1. *(a) An example of a class-graph. It is the* structure *of the graph that matters, not the nodes' letters. This structure, and the semantics of a class represented by a node, can be arbitrarily determined by the network administrator. (b) An example of a simple class-graph used in a network providing quality of service. Class A ≡ low latency and high bandwidth; B ≡ high bandwidth; C ≡ low latency; D ≡ best-effort.*

numerical value that indicates the number of hops the advertising node is from the destination. Another commonly-used class representation is the widest-path metric, which provides upstream nodes knowledge of the bottleneck downstream.

We note that, in general, it is possible to associate classes with representations that need not be directly comparable.[3] This necessitates another method of comparison, the basic requirement being the ability to indicate whether traffic of a particular class (say $A$) can be carried on certain routes (say of class $B$), or traffic of classes $A$ and $B$ cannot be carried on each other's routes. This inter-class relationship is represented by the class-graph, which consists of nodes, each representing a class of route or traffic, as well as directed links. A directed path $U \rightsquigarrow V$ exists between nodes $U$ and $V$ if class $U$ traffic can be carried on class $V$ routes. In this case, we say that class $V$ is *higher than* class $U$. For instance, in Figure 1(a), class $Q$ is higher than $D$. For classes of traffic that cannot be carried on one another, say $U$ cannot be carried on $V$ and vice versa, we say that these classes are *disjoint*. In Figure 1(a), classes $Q$ and $H$ are disjoint. The length, or number of hops, of a directed path does not have any significance.

If a cycle exists such that the directed paths $U \rightsquigarrow V$ and $V \rightsquigarrow U$ are present, this either means that (a) traffic of all classes in this cycle can be carried on routes of the same set of classes, which in turn implies that all nodes in the cycle can be collapsed into one, or (b) some policy violation has taken place. Grouping of classes in the case of (a) results in the class-graph being always directed and acyclic, thereby further reducing the complexity of operations performed on the graph. As a result, for any two distinct classes, one of them will be higher than the other, otherwise they are disjoint.

Different notions of classes, such as access privileges and quality of service, can be incorporated into the same graph, with the relationships subsequently defined using directed edges in the corresponding class-graph. This graph is used when computing route updates (§II-B), and is disseminated throughout the network with every router having the same graph. Since we do not expect the class-graph to change frequently, update costs should be low.

As an example, suppose the network administrator assigns the semantics to the following classes of routes: class A ≡ low

[3] In the sense that the relationship between a route of class 23 and another of class 27 doesn't have to depend solely on numerical comparison of "23" and "27".

latency and high bandwidth, class B $\equiv$ high bandwidth, class C $\equiv$ low latency, and class D $\equiv$ best-effort. The relationships between these classes are subsequently determined using edges in a class-graph, shown in Figure 1(b). It is easy to check that the resulting configuration is sane: best-effort traffic can be carried on links that incur low latency or have high bandwidth or both; traffic requiring high bandwidth cannot be carried on routes that guarantee low latency only; and traffic requiring low latency cannot be carried on routes providing best-effort service.

*2) Rules:* Route selection (RS) rules govern the selection of one route amongst multiple available ones. We propose two RS rules, either one but not both can be used in a network:

**RS1**: Given a set $R$ of incoming routes for a destination, select the subset $R'$ such that no route in $R$ is higher than any within $R'$. Using Figure 1(a) as an example, the set of classes $R'$ include $H$, $E$, $P$ and $Q$. Clearly, classes in $R'$ are disjoint. In the case of multiple equivalent routes from different neighboring routers, ties can be broken arbitrarily based on route preference, hop distance, router interface address etc.

**RS2**: Rather than selecting the highest classes, we propagate routes for all classes received. For a particular class, incoming routes eligible for consideration include those of higher and equivalent classes. Again, ties amongst routes can be broken arbitrarily.

In general, RS2 results in better distribution of traffic than RS1, which tends to load traffic of various classes onto routes of higher classes. The tradeoff is an increase in control overhead incurred when propagating routes' class information. The use of selection rules will be discussed later in §II-B, where we describe the routing process in detail.

*3) Transformation Boxes:* Currently, middle-boxes placed within a network affect the data and not the control plane. Functions performed by these boxes include firewalling, NAT, NIDS etc, and are also present in the transformation boxes (t-boxes). In addition, t-boxes are given the ability to manipulate the control plane by altering classes associated with routes propagated through them. Thus, t-boxes have a degree of control over the flow of traffic, and can complement that with data plane functionalities. For instance, to authenticate traffic of a certain class, a t-box can effectively channel that traffic through itself so that verification can be performed.

## *B. Routing On Classes*

In this section, we describe in detail how routing on classes is performed for link-state and distance-vector protocols, as well as for ethernets.

*1) Link-State (LS):* Link-state protocols propagate each edge's information throughout the entire network, allowing every node to maintain the complete network graph. The graph considered for a given class consists of all vertices and the set of eligible edges that are of higher or equivalent class. In general this means that the cardinality of the set of eligible edges decreases with higher traffic classes, an example of which is shown in Figure 2. In general, the subgraph can differ with the class of traffic under consideration. There are two ways to compute routes based on these graphs, corresponding to the route selection rules in §II-A.2.

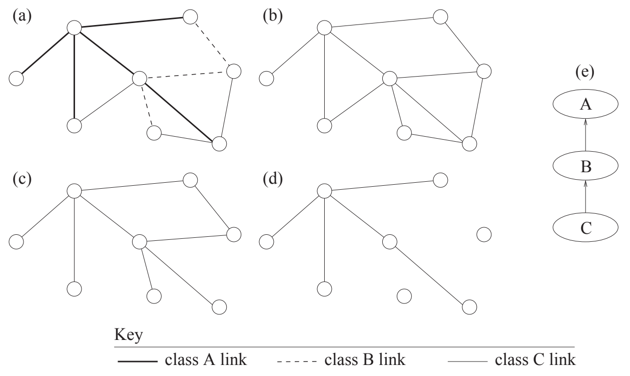

Fig. 2. *An example of reduction in set of eligible edges with higher traffic classes. (a) Original network graph showing classes associated with each edge. (b)-(d) Connectivity graphs used with classes C, B and A traffic respectively. (e) Class-graph for the network.*

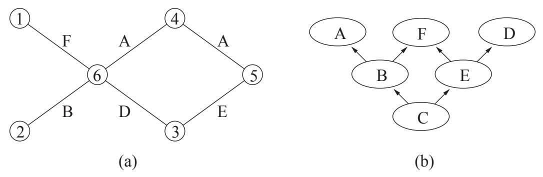

Fig. 3. *(a) Network using distance-vector routing and (b) class-graph in use.*

**LS1**: Each node performs shortest-path computation for every class of traffic on their corresponding connectivity graphs, obtaining the next-hop to the destination. A router uses an incoming packet's class and destination to look up the next hop.

**LS2**: An alternate scheme is to compute the shortest paths only for the highest classes (§II-A.1). For traffic of a particular class, the highest available class at that node is determined, and forwarding is performed on the corresponding subgraph. In this case, we trade a decrease in control traffic for an increase in per-packet processing time.

*2) Distance-Vector (DV):* These routing protocols propagate the cost of reaching a destination without revealing details of intermediate hops. Similar to link-state routing, both route selection rules result in slightly different class routing versions:

**DV1**: A router receiving advertisements for various classes computes the next hop for each class by selecting, amongst the subset of routes of equal or higher class, the route with the least cost. The next hops for each class are subsequently broadcasted to the router's immediate neighbors.

**DV2**: A router selects and uses routes only from the highest possible and disjoint classes.

Classes of links incident on a router need to be taken into account when disseminating routes: if the link class is lower than a selected route, then the highest class of route lower or

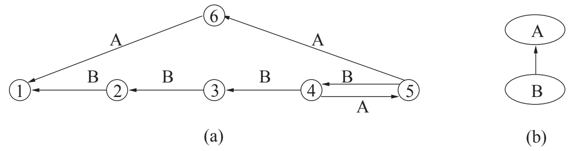

Fig. 4. *Class B traffic originating from node 5 and destined for 1 can oscillate between 4 and 5.*

equivalent to the link class is propagated instead. Clearly, this adjustment can be done at the neighboring router instead. For instance, assume that, in Figure 3, node 5 is the destination node and 6 receives advertisements with classes A and D from 4 and 3 respectively. Subsequently, 6 informs 1 of the availability of routes with classes B and E, and 2 of class B.

*3) Ethernets:* Ethernets are layer-2 networks utilizing an effective broadcast medium. In order to expand the coverage of an ethernet, bridges, or switches, are typically used, and spanning-tree protocols [4], [5] ensure that redundant bridging links are logically removed to eliminate forwarding loops for broadcast packets. Sapheniea in this scenario can be thought of as an extension to VLANs [6], which tag packets from end-hosts based on the ports they are attached to, or on their MAC address. The difference between the two is that in Sapheniea both end-hosts and ports are assigned classes. Packets received at destination ports are checked and dropped if the former's class is lower or disjoint. Thus, by altering the class of a port, an end-host will be able to receive packets from senders of different classes, not just from those of the same class.

*4) Loop-Free Guarantee:* Unlike traditional routing, which can be thought of as routing on a single class, loop-free routing on multiple classes requires guarantees in both the control and data planes. For the former, usage of shortest paths ensures that loops will be absent. The latter requires more thought: routing on classes can effectively result in multi-path routing. In this case, for traffic of a particular class, it can conceivably be carried on any route of equivalent or higher class. In Figure 4, traffic of class B originating from node 5 and destined for 1 can oscillate between 4 and 5 if the former chooses routes of class A and the latter, B, to forward the traffic. Two methods are available to guarantee loop-freeness:

**Monotonically-increasing most-recent class:** Packets received on a link can only be forwarded along routes with classes equivalent to or higher than that link's. Only knowledge of the previous link's class is required, no additional state has to be carried within the packet.[4] Using the same example in Figure 4, packets forwarded on class A links will not subsequently traverse class B links, thus eliminating oscillations.

**Decreasing distance to destination:** Each packet stores the previous hop's distance to destination, based on the class it was last routed on. The next hop chosen thus needs to satisfy both the class and per-packet distance requirement. It is easy to see that with a decreasing distance at each hop, the actual path traversed is guaranteed to terminate at the destination.

[4]Although depending on the router implementation, the previous link's class can be carried with the packet as meta-data.

## III. Enterprise Access Control

In this section we describe how Sapheniea can be used to ease management of access control within an enterprise. The discussion is applicable to large-sized networks using distance-vector or link-state routing, and also to small and medium-sized ones using ethernet. To make the discussion more concrete, we begin with the necessary steps taken by system administrators to realize access control within an enterprise, using Figure 5 as an example.

**Step 1, Traffic-to-class mapping:** We begin by mapping traffic to classes. This process is highly dependent on policies: accountants from the financial department are likely to be placed in a class (say $F$) disjoint from the research division's (say $R$). This mapping can be managed at a central location then disseminated to the routers, and is used to tag packets entering the network. Tags can also be dynamically determined via 802.1x [1], an authentication and key management protocol. Traffic not matching any entry can be assigned a default class.

**Step 2, Initial class-graph:** At this time, the class-graph should consist only of disjoint classes from the previous step. Directed edges reflecting relationship between classes are then added. For instance, superusers of class $S$ are given access to parts of the network open to either the financial department or the research division, thus directed paths $S \rightsquigarrow F$ and $S \rightsquigarrow R$ should exist in the class-graph. No additional classes are introduced in this step.

**Step 3, Link-to-class mapping:** Next, links in the network are assigned classes. In general, only links incident on restricted zones require careful assignment, remaining links can be set to allow the traversal of all traffic. In the case of ethernets, we can imagine that the routers and links at the center of the network form a common broadcast medium, with the links at the edges representing connections from end-hosts to ports (more details in §IV). A simple rule to decide on the assignment is as follows: (1) Select the highest and disjoint classes of traffic allowed on the link. If there is more than one such class, create another that is higher than all of them, adding the necessary directed edges. (2) Otherwise the link's class is set to be that of the highest traffic class.

Note that both the forward and reverse paths between two end-hosts should be taken into account. In our example, packets from superusers are tagged with class S, and can traverse class A links (ports) that researchers are connected to. Replies from researchers on the other hand are tagged with R, and in this case the link (port) to which superusers are attached have sufficiently high class for them to traverse. Alternatively, if superusers are attached via a link of class S, per-flow state can be stored at the researchers' end-router, allowing that router to promote the class of returning packets from R to S.

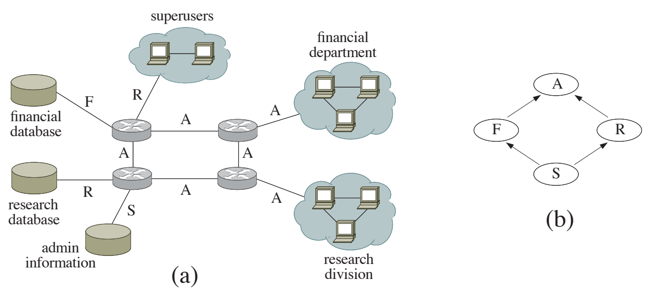

Fig. 5. *Enterprise access control: (a) Link class assignment providing the required access control using the class-graph in (b), where accountants' packets are tagged with F, researchers' with R, and superusers' with S.*

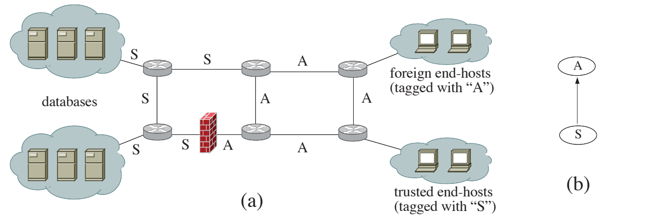
databases
foreign end-hosts (tagged with "A")
trusted end-hosts (tagged with "S")
S
A
(a)
(b)

Fig. 6. *Traffic channeling: (S ≡ secure traffic only, A ≡ all traffic). By altering classes of routes to the databases as they propagate, the firewall is able to channel foreign traffic destined for them through itself. Once verified (and not dropped), foreign traffic's class is converted, thereby allowing it to traverse secure parts of the network.*

Figure 5 shows the final class-graph and link class assignments, it is easy to see that (1) packets from superusers and researchers can reach one another, as can those from accountants and researchers, but packets from accountants cannot be forwarded towards superusers; (2) superusers have access to all parts of the network, including the financial and research databases, as well as administrative information; and finally (3) the financial database can be accessed only by superusers and accountants. Similarly, the research database is accessible only by researchers and superusers.

## IV. Traffic Channeling

Perimeters around restricted zones can be set up within a network to control access to them. The major problem lies in ensuring that there are no unchecked data paths, which is difficult since these can exist because of wireless links, or even because of portable media that was infected with worms [15]. We attempt to reduce instances of unintentional access by minimizing configuration errors when constructing perimeters.

A step in this direction is the removal of dependence of middle-boxes on the physical topology. Rather than determining all possible data-paths and placing choke-points on all of them, we instead prefer directing traffic through them. Transformation-boxes (t-boxes) are middle-boxes with typical data-plane functionalities such as NATs and firewalls, but are also endowed with the ability to manipulate the control plane. The latter is achieved by altering the class of a route propagating through it: raising a route's class allows the forwarding of more classes of traffic. In Figure 6, the firewall

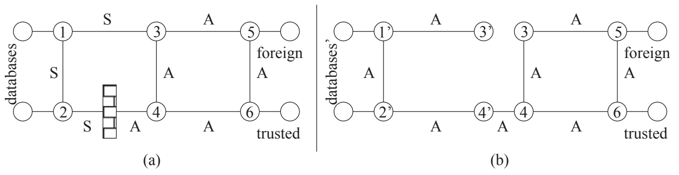

Fig. 7. *Link-state network graph (from Figure 6) manipulation: (a) the t-box, denoted by the firewall symbol, translates links of class S to A by (b) creating a logically different part of the network corresponding to the class of traffic it can forward. It also inserts an additional link (4', 4) bridging the two parts.*

converts classes of routes terminating at the databases from S to A, thereby allowing foreign hosts access to them but only via the firewall. Below, we describe how routing protocols can be altered to enable this.

### A. Link-state

Since every router is aware of the entire network graph, enabling route transformation in link-state routing requires some thought. Two methods can be used, we assume that class S routes are to be transformed to class A:

**T1, Propagation of additional link information:** For each advertised link of class S, the t-box generates an additional link of class A, as well as logically different nodes at the end-points of that link. Furthermore, a link representing the firewall is added between the two parts of the network. Figure 7 gives an example of this.

**T2, Consideration during local computation:** Each router in the network is notified of the t-box's presence. When computing the shortest path to each destination, the class associated with the current route is altered accordingly.

T1 has the advantage that routers need not be aware of its presence and do not require T2's special route computation ability. The tradeoff is an increase in bandwidth consumption for disseminating additional link information, as well as memory for storing them.

### B. Distance-vector

Since distance-vector routing hides details of the network path chosen, no other changes to the protocol is required: a chosen route is simply propagated with the new class.

### C. Ethernets

In ethernets, the route between source and destination is determined dynamically, by storing the next hops from which the broadcast ARP [11] query and resulting unicast reply arrive. We tweak this process slightly, and show that traffic can be directed to flow through our t-box. We step through the process below, using Figure 8 where node S is the sender (class A), R the receiver (class B), and T is the t-box, and denoting a sender X with class Y and port Z by X.Y:Z.

**Step 1, Broadcast from S:** The query packet is tagged with S' and S' outgoing port's class (both A) as it enters the switch. In addition to storing the incoming port and MAC address of the broadcast query, the switch also store the sender's and port's class. Since the query (class A) will be dropped at R's port (class B), R will not respond to it.

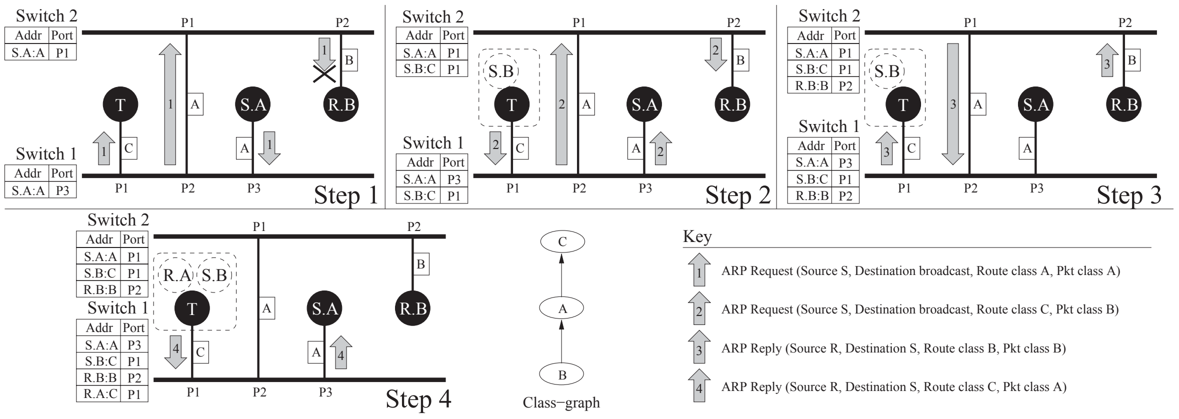
Switch 2
Addr Port
S.A:A P1
S.B:C P1
R.B:B P2
R.A:C P1
Switch 1
Addr Port
S.A:A P3
S.B:C P1
R.B:B P2
R.A:C P1
P1
P2
P3
T
S.A
R.B
S.B
R.A
Step 1
Step 2
Step 3
Step 4
Class−graph
C
A
B
Key
ARP Request (Source S, Destination broadcast, Route class A, Pkt class A)
ARP Request (Source S, Destination broadcast, Route class C, Pkt class B)
ARP Reply (Source R, Destination S, Route class B, Pkt class B)
ARP Reply (Source R, Destination S, Route class C, Pkt class A)

Fig. 8. *The t-box uses the broadcast and address learning mechanisms of ethernet to effectively assume the end-host roles in alternate class domains.*

**Step 2, Rebroadcast from T (S.B):** T reacts to the query, with the response being triggered either by looking up R's IP and class,[5] or after some timeout period if no response is heard from R. T rebroadcasts the same packet, this time with R's class (if known), or otherwise with a special broadcast class that can be accepted by all classes of ports. Thus, T effectively takes on the role of S.B. As the rebroadcast packet travels through the network, switches store the previous hops to S.B. Now the query (with its class removed upon exiting the network) will be able to reach R, which subsequently replies with its MAC address.

**Step 3, Reverse path to T (S.B):** Since the ARP reply is addressed to S and is of class B, it can either be forwarded towards T or S (which does not violate policy). However, to prevent loops (see Step 4), we look up the exact address and class match and thus forward the packet to T, where its class is changed to A after which it is sent back into the network.

**Step 4, Reverse path to S.A:** Since the packet can be forwarded back to S.B (*i.e.* T), we mandate sending to the port associated with the address and class that exactly matches the packet's destination and class. If no such entry in the MAC table exists, then a consistent method of selecting next hop ports should be used. Also, T's port should be assigned a class different from any hosts', otherwise the network may end up with two S.A entities. Once these conditions are met, no loops will exist and the packet reaches S.A.

After this initial learning phase, all subsequent packets between S and R will traverse T.

## V. Related Work

SANE [2] is an enterprise-level security architecture that uses a centralized domain controller to provide routes and capabilities to for each connection in the network. SANE requires substantial changes to the network, such as link advertisements instead of the commonly-used spanning tree protocols, and may not scale up to an intra-domain level.

Predicate routing [12] is similar to Sapheniea in that links are associated with predicates defining the kind of packets allowed, then a suitable path for a flow satisfying all conditions is picked. Thus, routing and firewalling are integrated. However, because of the need to know all the predicates along a path, it does not work with distance-vector protocols which hide details of the path beyond the next hop, and computation resource consumption can be significant since the routing is performed on a per-flow basis.

[5]Thus necessitating existence and configuration of the map in T.

## VI. Summary

We introduced the Sapheniea framework, which provides (1) a simple, graphical way of representing access control policies in the network using class-graphs, (2) routing based on classes, eliminating the need to reinstall filters when topology changes, and (3) transformation boxes, which are middle-boxes with the ability to alter routing and thus traffic flow. We believe that Sapheniea will significantly reduce configuration errors and improve the manageability of networks.

## References

[1] L. Blunk and J. Vollbrecht. RFC 2284: PPP extensible authentication protocol (EAP), Mar. 1998.
[2] M. Casado, T. Garfinkle, A. Akella, M. J. Freedman, D. Boneh, N. McKeown, and S. Shenker. SANE: A protection architecture for enterprise networks. In *15th USENIX Security Symposium*, Aug. 2006.
[3] W. Cheswick and S. Bellovin. *Firewalls and Internet Security*. Addison-Wesley, 1994.
[4] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks - Common Specifications Part 3: Media Access Control (MAC) Bridges, 1998.
[5] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Local and Metropolitan Area Networks - Common Specifications Part 3: Media Access Control (MAC) Bridges - Ammendment 2: Rapid Reconfiguration, June 2001.
[6] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE 802.1Q, IEEE Standards for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks, 2003.
[7] G. Malkin. RFC 2453: RIP version 2, Nov. 1998.
[8] J. Moy. RFC 1247: OSPF version 2, July 1991.
[9] D. Oran. RFC 1142: OSI IS-IS intra-domain routing protocol, Feb. 1990.
[10] V. Paxson. Bro: a system for detecting network intruders in real-time. *Comput. Networks*, 31(23-24):2435–2463, 1999.
[11] D. C. Plummer. RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, Nov. 1982.

[12] T. Roscoe, S. Hand, R. Isaacs, R. Mortier, and P. Jardetzky. Predicate routing: enabling controlled networking. *SIGCOMM Comput. Commun. Rev.*, 33(1):65–70, 2003.

[13] Snort.org. Snort, the open source network intrusion detection system. `http://www.snort.org`.

[14] P. Srisuresh and K. Egevang. RFC 3022: Traditional IP network address translator (traditional NAT), Jan. 2001.

[15] N. Weaver, D. Ellis, S. Staniford, and V. Paxson. Worms vs perimeters: The case for hardLANs. In *Hot Interconnects*, Aug. 2004.

[16] G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, and G. Hjlmtsson. Routing design in operational networks: a look from the inside. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 27–40, New York, NY, USA, 2004. ACM Press.

[17] G. G. Xie, J. Zhan, D. A. Maltz, H. Zhang, A. Greenberg, G. Hjalmtysson, and J. Rexford. On static reachability analysis of IP networks. In *Infocom*, 2005.