# Simplifying Access Control in Enterprise Networks

*Cheng Tien Ee*
*John Lee*
*Dave Maltz*
*Scott Shenker*
*Lakshminarayanan Subramanian*

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-33
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-33.html

March 11, 2007

# Simplifying Access Control in Enterprise Networks

Cheng Tien Ee[†], John Lee[‡], Dave Maltz[§], Scott Shenker[†♮], Lakshminarayanan Subramanian[‡]
[†]*UC Berkeley*, [‡]*NYU*, [§]*Microsoft Research*, [♮]*ICSI*

## Abstract

Today, access control configuration in large enterprise environments is a highly complex process that involves the manual configuration of a wide range of network devices including routers, VLANs and firewalls. Much of this complexity arises from the asynchrony between routing and access control that often requires contorted network topologies that lack redundant paths, have tight pinning of routes, and physical placement of firewalls along the data path to achieve access control.

In this paper, we propose *Access Control Routing (ACR)*, a clean-slate and flexible approach to simplify access control configuration in large-scale enterprise networks. ACR uses a single parameter, *class*, to couple access control and routing. It requires that each end-host specify its access control policies at the granularity of a class. On the network side, the control plane establishes logical reachability networks for every class, and the data plane explicitly labels each packet with a class based on the source. Unlike traditional access control configuration approaches, ACR can easily adapt to network topology or routing changes and is better suited to handle network failures. ACR eliminates the need for VLANs and also provides the flexibility of automatically routing traffic through arbitrary middle-boxes without physical topology manipulation. Using a software-based router implementation of ACR and access control policies gathered from four large commercial enterprise networks, we show that ACR can easily be adopted in large enterprise environments with little additional performance overhead.

## 1 Introduction

Enterprise networks play a critical role in the security policies of the organizations they serve by enforcing *access control*. Access control typically involves two components (1) blocking packets between hosts that are not allowed to communicate at all, and (2) ensuring that a host does not receive or send packets unless they are first passed through a middle-box where they can be inspected, audited, or scrubbed.

Many techniques exist for implementing access control [1, 3, 7, 16, 17, 23, 24]. However, setting up and operating these systems requires maintaining the consistency of a tremendous amount of state that is distributed across all the components of the network. For example, the rules that define which hosts can communicate are typically written in terms of hosts' IP addresses, yet the middle-boxes or firewalls where these rules must be installed can be located anywhere in the network, with no obvious or direct relationship to the hosts. Any change to the IP address assigned to a host invalidates the rules, but the burden of ensuring consistency falls to the network administrators. An example of maintaining the consistency of state would be the BGP routing policies used in some enterprises to enforce coarse-grained reachability policies, and the physical manipulation of links to ensure packets traverse a particular middle-box. The tremendous amount of state that has to be considered simultaneously makes the network brittle and configuration error-prone.

This paper proposes and evaluates *Access Control Routing* (ACR), a way of using the routing protocols themselves to distribute the state needed for access control, thereby automatically maintaining consistency and reducing configuration. Our approach to network access is inspired by the model of file access control, where users belong to groups and, in turn, groups are permitted or denied access to files.

In ACR, administrators define *classes* in their network. For any entity in the network, administrators can specify the classes that entity can send packets to and the classes it can receive packets from. We use the routing protocol to create logically separate networks for each class, so that if a host is not allowed to send packets to a particular class, none of the destinations in that class will even appear routable. Packets are explicitly marked with the class they are part of, but middle-boxes are able to change the class markings on packets and re-advertise destinations from one class to another. This primitive creates an easy and flexible way to channel packets through middle-boxes.

ACR has four major benefits: (1) Configuration is simplified because the designer needs only group the appropriate end-hosts based on their roles, and thereafter the routing protocols themselves maintain the resulting reachability and path constraints. (2) ACR provides a flexible framework for channeling packets through any number of "bump-in-the-wire" firewalls [3, 7] and deep-

packet inspectors [15,16] in any order desired. (3) ACR frees administrators to assign IP addresses based on network topology rather than what will ease writing packet filters, since filters will no longer be written in terms of addresses but rather classes. (4) ACR retains the advantages of network-level access control over pure host-based access control in that Denial-of-Service attacks or worms that might overwhelm or subvert a host can be discarded before they even reach the host.

In this paper, we first motivate our work using high-level observations of deployed networks in §2. In §3 we motivate the basic idea of how configuration using classes and describe its benefits. Later in §4, we describe the ACR design in detail. Implementation details are given in §5 and we argue that the modifications involve only slight tweaks. In §6, we discuss our implementation of ACR on the Click modular router [12], as well as provide analysis of four large commercial networks. Using a combination of qualitative and quantitative analysis, we show how ACR will reduce configuration and administrator burden. Also, we show that ACR can easily handle the access control requirements of existing large enterprise networks. Since ACR is clearly reminiscent of *virtual LANs* (VLANs) and other network virtualization techniques like MPLS, we explain the differences between ACR and these approaches, as well as other related work, in §8.

# 2 Networks in Practice

To better understand the issues with current networks and to facilitate comparison later, we gathered topology and configuration information from two large intra-domain networks: a large commercial enterprise (ComNet) that interacts with a variety of client entities and a university campus network (UNet). These networks follow two general models, which we call *core* and *edge*. We begin by describing the similarity between these two models, followed by an elaboration on the differences between them.

## 2.1 Aggregation of Hosts via VLANs

In both core and edge models, hosts that should have the same reachability policies may be plugged into different switches. To ensure these hosts all receive IP addresses from the same subnet so that IP routing policies and packet filters can be applied to them as a group, designers are forced to drag VLANs through multiple switches to gather each set of related hosts into a single virtual layer-2 LAN before connecting them to a layer-3 router. Configuring these VLANs (which are essentially multi-point permanent virtual circuits) is a painful and often manual process, requiring designers to carefully assign the order in which switches will become the root bridge and explicitly prune links to ensure the spanning tree protocols behave reasonably [4].

At layer-3, configuration methodologies between these two types of network begin to diverge, in terms of routing and middle-box set up. We next elaborate on the two models, focusing on the differences between them. We begin with the ComNet network.

## 2.2 The Core Model: A Commercial Network

As shown in Figure 1a, the commercial network is comprised of two main regions: an external MPLS network that provides connectivity to various client organizations, and an internal network that hosts various services. A demilitarized zone (DMZ) sits between these two network partitions, filtering unwanted packets from the external network.

The internal network consists of VLAN subnets at the periphery, and these are connected via OSPF [13]. At the center of the network, core routers running BGP [19] restrict the flow of packets in a coarse-grain manner using import and export policies. Firewalls are co-located with core routers, siphoning off packets for inspection and re-injecting them via alternate ports.

ComNet epitomizes the Core Model (CM), which has the following characteristics:

**Few, Heavy-Weight Middle-boxes:** A relatively small number of middle-boxes are placed at locations traversed by most traffic, such as within the network core region. Since packets can be sent from any source and destined to any end-host (as opposed to edge routers that see packets destined for or sent from its subnet), firewall rules tend to be numerous resulting in an increase in state required and processing delays. Furthermore, such rules are often difficult to debug, and rely to a large extent on the routing protocol, which ultimately dictates which traffic flows traverse the middle-boxes. On the other hand, having fewer middle-boxes may ease configuration time and complexity.

**Controlled Routing:** The second characteristic of CM is that routing, or more precisely the actual paths taken, is explicitly configured. In the case of ComNet, the export and import rules of BGP, which in ISPs are used to reflect economic policies, are enlisted here to enforce this control. In addition to BGP, the physical configuration of the network has to be restricted as well, as exemplified by the single link through the DMZ to ensure all packets traverse the two firewalls there. In general, currently available means for implementing security push designers towards static routing that is particularly prone to network link failures, resulting in brittle networks that lack redundancy. Furthermore, addition

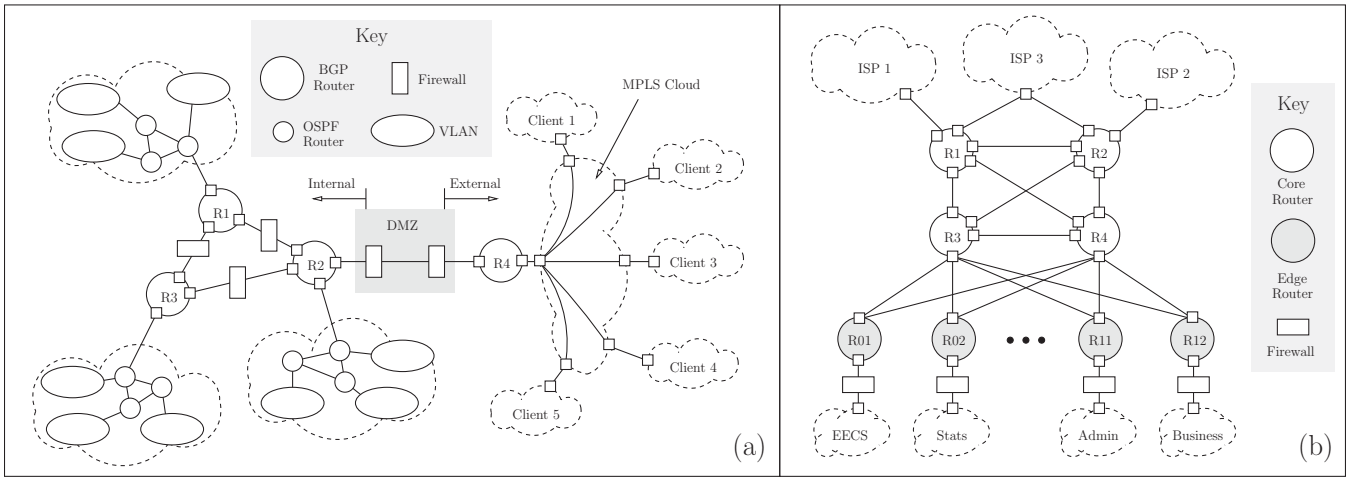


Figure 1: (a) General layout of the ComNet commercial network, where firewalls are placed in the core of the network, and routing is constrained for traffic to traverse these firewalls. (b) The UNet university campus network structure, firewalls are usually placed at the edges, next to the departmental subnets, and routing is unconstrained.

of new links can unintentionally result in the availability of alternate paths that bypass middle-boxes [6].

### 2.3 The Edge Model: A Campus Network

Figure 1b gives the high-level view of the UNet campus network. Since this is primarily a network that caters to learning and research purposes, the interior of the network has no access restrictions. As such, a single intra-domain routing protocol, OSPF, is sufficient without requiring BGP. The network itself serves multiple departments, which may or may not utilize the firewalls placed at the edge close to the departmental networks. The UNet network is representative of the Edge Model (EM), which has the following features:

**Multiple, Light-Weight Edge Middle-boxes:** Middle-boxes are placed as close to the client subnets as possible, allowing them to handle smaller destination and source sets thus reducing the state and number of rules required. However, the total number of middle-boxes to configure is therefore higher, increasing configuration complexity. On the up side, unintentional bypassing of middle-boxes is less likely to occur since it is easier to control the physical connections to the subnets.

**Unrestricted Routing:** The advantage of placing middle-boxes at network edges is that fewer constraints are needed on routing, since the physical chokepoint makes it harder to bypass the middle-boxes. Consequently, these networks allow additional links to be added, with the routing protocol automatically adjusting paths to take the new links into account.

### 2.4 Summary of Differences

In summary, current network designers either control routing to constrain traffic paths, resulting in brittle networks, or push the complexity to edges of the network, thereby necessitating the configuration of more middle-boxes. In some instances, BGP's route import and export ability has been enlisted to provide the necessary level of path control. In others, alternate paths are eliminated to produce physical chokepoints that channel the traffic. Unfortunately, both approaches decrease path diversity, making the network failure prone and increasing recovery times.

Additionally, high-level access policies need to be translated manually into the form of rules and installed at middle-boxes. The difficulty in translating between high-level policies and low-level rules means that numerous firewall rules are difficult to make correct. From anecdotal evidence, the lack of coupling between routing and access control at the protocol level can be further compounded by the fact that they may not be under the same administration. In the next section we describe our design, and show how the entire configuration process can be simplified.

## 3 Configuration Using Classes

In this section, we describe how the basic notion of *classes* in ACR can be used to simplify configuration in enterprise environments. We begin by describing the configuration interface followed by a brief description of

how ACR uses class information to achieve access control. Finally, we summarize the benefits of class-based configuration.

## 3.1 ACR Configuration Interface

The fundamental idea behind ACR is to use the abstract notion of classes to simplify enterprise configuration by establishing logically distinct networks corresponding to each class. In ACR, sources and destinations specify access control policies using classes, and traffic that flows through the network is categorized into different classes. A destination host or service specifies an *access policy* that states the classes of incoming traffic it will accept. Thus, configuration of a source's access control policies is reduced to specifying the class(es) of traffic allowed to originate from that source.

Table 1: Access Policy Assignment

| Host | Can Receive Packets of Class |
|---|---|
| Researcher | R, A |
| Financial Department | F, A |
| Database 1 | D, A |
| Database 2 | E, A |
| Administrator | A |

Table 2: Class Membership Assignment

| Host | Can Send Packets of Class |
|---|---|
| Researcher | D,E |
| Financial Department | D |
| Database 1 | R, F, A |
| Database 2 | R, A |
| Administrator | A |

We motivate this using a simple example. Consider a network where we have two databases 1 and 2, and three entities who wish to access the databases: a researcher, end-users in the finance department and the administrator. While the researcher needs access to both databases, end-users in the finance department are given access to only database 1 and the administrator can access all the resources in the network. For this example, Tables 1 and 2 specify the *access policy* configuration for each destination and the *class membership* configuration for every source respectively.

We make several important observations. First, specifying access control configuration using classes is trivial both for traffic sources and sinks. From an administrator's standpoint, a class represents an abstract category of traffic on which the administrator can set access and class membership policies. In practice, a class can refer to a particular network service, an end-host, or the current state associated with the packet as it makes its way across the network. Therefore, the translation from access control rules to class assignments need not be unique.

Next, for any given class, it is easy for an administrator to visualize as well as manually verify its configuration settings. Furthermore, all hosts that have similar access control requirements can be grouped into a single class *e.g.* class F for the finance department. This is critical to reduce the number of classes required in the network. Finally, class assignment need not be symmetric; in the example above, packets from researcher to database 1 are marked as class D, and those in the reverse direction are marked with R.

## 3.2 How Does ACR Work?

Considering the same example, we next describe how ACR translates class-based policies to achieve access control. We have again the case of databases 1 and 2, which the administrator determines are able to accept packets of class D and E respectively (Table 1). This information is installed in the router(s) to which the databases are connected, and it is propagated by the routing protocol. In the control plane, routers compute *reachability on a per-class granularity*. For example, if the underlying routing protocol is link-state routing, then the link to database 1 is labeled class D and this information is used in the route computation process. Therefore, all routers within the network will establish class D and class E routes to databases 1 and 2.

Next, suppose the researcher is to be given access to the databases. The class membership configuration for a host is stored in its first-hop router. When the researcher's machine generates a packet destined for database 1, the first-hop router examines the class membership list to determine if there exists any class in this list for which a class-based route to database 1 exists. Upon finding a matching class (class D), the first-hop router marks the packet as class D and forwards it along a class-D route. Every intermediary router examines the class header and forwards it along the corresponding class route. At the last-hop router, the label is removed, thus eliminating the need to alter either the researcher's or databases' machines.

Hence, ACR explicitly binds access control with the routing protocol and establishes routes on a class granularity that is in sync with reachability constraints. The role of the data plane is simplified to a straightforward verification of whether the class membership list of the source matches with any of the available class-based routes to the destinations.

## 3.3 Why Classes?

Apart from simplifying access control configuration, distinguishing packets based on logical classes, rather than physical connectivity, provides the following properties:

**Network End-to-End Visibility:** Since the destina-

tions reachable by a packet are determined by its tag and enforced by routers, a packet that does not have the necessary permission to traverse the logical network in which the destination resides is dropped immediately. Verification is performed as part of the forwarding process, and takes place at all hops along the path.

**Route Redundancy:** The mechanism to check eligibility of access has shifted from traffic channeling and packet header inspection at firewalls to classification at first-hop routers and verification at all intermediate ones. The usage of classes as a mechanism to enforce access has therefore become orthogonal to the routing process. As a result, it is no longer necessary to constrain path diversity in order to channel packets through middle-boxes, and the network designer is free to add any number of links at any place in the network.

**Topology-Independent Middle-box Placement:** One of the distinctive features of the edge and core models discussed earlier in §2 is the placement of the middle-boxes. The locations of these boxes must be carefully chosen to align with the routing protocols, routing design, and physical topology of the network so that the desired set of packets traverse them. Otherwise, the middle-boxes have to be placed at every point along the network edge.

Current networks are designed with the physical dimension in mind: for a source A and destination B, the middle-box has to be placed along the path A$\rightsquigarrow$B. In ACR, we think in terms of the class dimension: for a source class A and destination class B, the middle-box has to bridge the two domains. In other words, the middle-box forms the "path" between the two points, and if it is the only path, then packets traversing between the two class domains must go through it. Thus, the actual network links traversed, as well as the physical location of the middle-box, no longer matter.

**Ease of Understandability:** Last but not least, since the network designer works with high-level policies in place of low-level details, the required inputs to the system are easier to understand. This eases the transition phase when another administrator takes over, or when changes need to be made to the network, hence reducing the likelihood of errors.

# 4 ACR Design

In this section, we describe how we realize this abstract notion of multiple logical networks in ACR. Bearing in mind the conditions commonly encountered in networks today, we begin by stating our design space that provides scope and allows us to focus on the most relevant issues. Then, we describe the ACR control and data plane operations in detail.

## 4.1 Design Space and Assumptions

As mentioned earlier, fine-grain access control to individual objects is best implemented on the hosts themselves in application-specific ways, such as through Kerberos [14], TLS [5], or file Access Control Lists (ACLs). At the network-level, however, designers may wish to have a class represent an entire organization, or a particular user, with the decision dependent on the nature of the policies they are trying to implement. Our solution provides that flexibility, and does not place any constraints on actual class assignment.

Next, we assume that routers can be trusted, that physical access to them is restricted and that they are not compromised. These assumptions are all true in typical enterprise networks, though security measures, such as the cryptographic authentication option in OSPF2 [13], can be used to reduce further the probability of successful attacks. In general, routers share fate and thus the bringing down of one is likely to significantly and negatively impact the entire network.

We believe that any solution to network-level access control must not depend on changes to hosts connected to the network, since these hosts may not be under the control of the network administrators. Further, one of the benefits of network-level control is having an independent line-of-defense even when hosts are compromised.

## 4.2 The Control Plane

We begin with the control plane, operations of which include (1) the assignment of classes to hosts, (2) dissemination of access control information by the routing protocol, and (3) installation of classification information at first-hop routers.

As a result of these operations, the control plane establishes the following state in each router. (1) For each destination reachable from that router, the latter will also know what classes the destination is willing to accept. (2) Each router with directly-connected hosts knows the classes each of them are allowed to send. §4.3 explains how the first-hop router uses this information to label the packets sent by a directly-connected host with the correct class.

### 4.2.1 Assignment of Classes

The network designer can implement static policies that specify which entities on their network should be allowed to communicate. A network entity be any of the

following: an end-host (defined by MAC or IP address), a group of hosts (defined by IP prefix), or a well-known service (defined by port number). For each entity, the administrator defines the classes the entity is allowed to use when sending packets (the entity's *class membership*) and the classes of the packets the entity is allowed to receive (the entity's *access policy*).

As an example, suppose we have the following end-hosts: a researcher, the financial department, databases 1 and 2, and an administrator. Tables 2 and 1 show the classes each host can send to and receive packets from respectively. We see that the researcher's packets are tagged with class D and E, thus allowing them to reach databases 1 and 2, but not the financial department nor administrator. In general, since communication is typically bi-directional (*e.g.* via TCP) we require access to be *symmetrical*, that is, we expect the sender to be able to receive response packets. Thus, databases 1 and 2 can send class R packets that can reach the researcher. However, a class need not be symmetric: just because a host can send packets of class A does not necessarily imply it is allowed to receive packets in class A. For example, by the policy in Tables 2 and 1, two hosts in the financial department are not allowed to communicate with each other.

#### 4.2.2 Class Dissemination via Routing

Routing protocols, either link-state, distance-vector or path-vector, need only be slightly modified to account for the use of classes (details in §5). Access policies associated with end-hosts are installed at their corresponding first-hop routers and disseminated by the routing protocol in use. For the example in Table 1, the subnet to which financial department hosts belong will be advertised with classes F and A.

Next, the routing process is carried out for each class, with intermediate routers storing corresponding forwarding information for each. One of the advantages of integrating routing with access control is the ease with which traffic channeling can be performed. This is important, as middle-boxes that perform deep-packet inspection for worms *etc.*, firewall filtering, or statistics gathering are not useful unless the target traffic is routed through them.

While channelling traffic by physically manipulating network connections may sound simple and straightforward, it is much more complex at the ground level, where machine rooms are often filled with intertwined cables plugged into a multitude of switchers. This complexity increases the likelihood of errors — for instance, the administrator may inadvertently add a link and thus unintentionally allowing traffic to bypass the firewall.

ACR achieves traffic channeling through the concept of *class transformation*. The middle-boxes, which we call *class transformation boxes* or t-boxes in short, alter the classes associated with destinations as routes are propagated. Data packets' class tags are also altered accordingly as they are forwarded through these t-boxes. For example, servers that should only receive packets that been through a scrubber t-box will accept packets of class $after\text{-}scrubber$. T-boxes that offer the scrubbing service are the only devices allowed to send packets of class $after\text{-}scrubber$. When the t-box receives a route to destination in class $after\text{-}scrubber$, it reannounces the destination in class $before\text{-}scrubber$. Hosts wishing to contact the server then send packets to class $before\text{-}scrubber$.

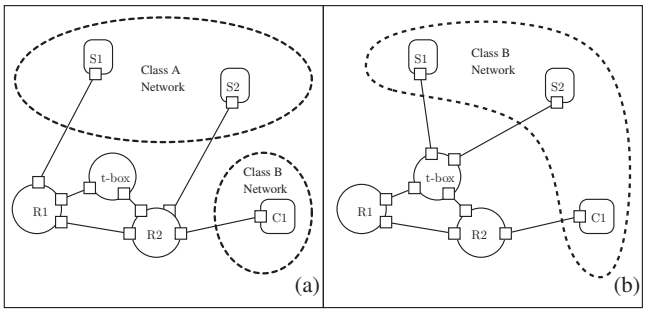


Figure 2: T-boxes translate packets between classes. (a) Original access configuration: servers S1 & S2 are in class A, client C1 in class B. (b) (from viewpoint of C1) By re-advertising reachability of S1 and S2 into class B, the t-box puts itself on the path for reaching S1 and S2. Packets from C1 to S1 cannot avoid the t-box, even though alternate physical paths exist, because the only route in Class B to S1 comes from the t-box.

For link-state routing, t-boxes re-advertise reachability of destinations in the new class(es). This is similar to area border routers in OSPF [13], and is illustrated in Figure 2, where in (a) one can view the classes as separate logical networks, and (b) from the viewpoint of hosts in a particular network, those in the other are reachable only via the t-box. Thus, there can be multiple t-boxes placed in the network, providing robustness without reconfiguration if any one fails.[1] However, as before, if both directions of a flow has to traverse a t-box, say for reasons of completeness necessary for flow analysis, then introducing multiple t-boxes with the same functionality may cause the flows to become asymmetrical.

[1] We expect many t-boxes will be stateless, such as packet loggers or inspectors. Even if t-boxes are stateful, such that the failure of a t-box or rerouting means that connections through the t-box will need to be restarted, our proposed system is still more reliable overall that today's networks, where the failure of a t-box typically partitions the network in a manner that end-hosts cannot recover from at all.

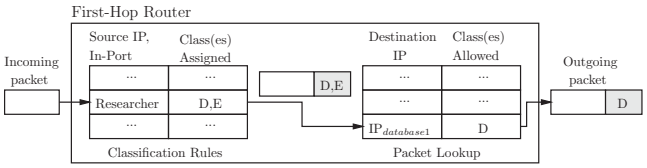


Figure 3: Packet forwarding at the first-hop router. The packet is first classified based on source IP, incoming interface port, *etc.*, then the intersection set $\mathcal{I}$ of assigned classes and classes permissible at destination is determined. The packet is dropped if a null set results, otherwise it is tagged with one class in $\mathcal{I}$ and forwarded.

#### 4.2.3 Class Information Installation

Given the current practice in enterprise networks, we expect that most network designers will choose to assign the classes a host can send and receive by configuring this information into the interface/port on the first-hop router when the host is connected. Alternatively, networks that already inventory their hosts' MAC and/or IP addresses might choose to configure classes based on this table.

### 4.3 Data Plane

In our framework, a host's first-hop router (that is, the first router its outgoing packets arrive at) is responsible for labeling the host's packets with an appropriate class marking. We choose this approach for three reasons: (1) thanks to the distribution of class information through the control plane, routers have the required information to know which classes a packet's destination will accept, (2) marking packets at the routers eliminates the need for changes at the hosts, and hosts typically outnumber routers by almost 100 to 1; and (3) routers are assumed to be trusted.

The classification and forwarding of data packets at the first-hop router is shown in Figure 3. We begin with classification based on source IP, the interface port the packet arrived on, *etc.*. A packet may be tagged with multiple classes at this time, representing permissions to access various resources. Next, we look up the next hop interface using the destination IP address. If the destination is reachable, the routing table will contain both the next-hop (as normal) and the set of classes the destination accepts (as established by the control plane extensions described in §4.2).

If the intersection of the tagged and permissible classes results in a non-null set, the packet is forwarded after inserting in its header one of the classes in the intersection set. Otherwise the packet is deemed not to have permission to reach the destination, and is dropped. For intermediate routers other than t-boxes, no additional classification needs to be performed. Instead, we simply check that the destination is able to accept the traffic class carried by the packet. Finally, as noted before, packets traversing t-boxes may have their class tags altered before being forwarded.

### 4.4 Using ACR in Practice

In the sections above we described the core framework and mechanisms of ACR. This section illustrates how these mechanisms are sufficiently general to handle deployment scenarios that arise in enterprise networks, both common and uncommon ones.

#### 4.4.1 Connectivity to External Networks

ACR provides an easy mechanism to control reachability to external networks. The designer configures the border routers to announce all external routes into the enterprise network with a class *external*. Hosts can then be granted reachability to the outside world simply by giving them permission to send and receive packets with class *external*. In general, there is no longer a need for a rigidly structured DMZ as in the CM model. If designers are concerned about DoS traffic entering their internal networks and congesting links, they still have the option to place firewalls near their borders as described next.

#### 4.4.2 Stateful Firewalls

A very common middle-box in today's networks is a stateful firewall that allows packets into a network only if they are associated with packets that have previously been sent out of the network. The intuition is that the packets sent from inside the network constitute an invitation for the response traffic, and only the response traffic, to enter the network. In the most common incarnation, TCP packets are allowed from outside to inside only if they belong to the same flow as a TCP SYN sent from inside to outside. Some such middle-boxes are also network address translators, modifying the addresses of the packets as they flow through, but this is orthogonal to the stateful firewalling.

Under ACR, the logic inside the middle-box remains exactly as it is today, and they can NAT or not as they choose. The only difference is that under ACR the middle-box will change the class of packets as they flow through. In a typical deployment, the administrator will define classes *internal* and *external*, with the hosts to be protected by the firewall sending and receiving packets of class *internal* and the border routers handling *external* packets as described above. The middle-boxes can now be placed wherever desired in the enterprise network, and the appropriate traffic will be routed through

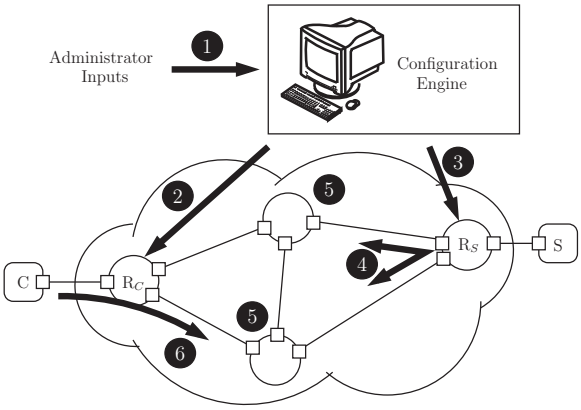


Figure 4: High-level view of configuration and automated network-level operations supporting ACR. C is the client machine that intends to communicate with server S. $R_C$ and $R_S$ are the first and last-hop routers respectively.

them. Even though there is a single *internal* class, packets from the outside cannot go to a internal host that does not expect them, as the existing logic will ensure that the packet is part of an established and desired flow before mapping it from *external* to *internal*.

#### 4.4.3 Eliminating VLANs

Networks using ACR should no longer need VLANs, as classes serve the same function. Where a designer would have created a VLAN to separate traffic on their network, they can now create a class. As explained in §5, ACR can run over link-state, distance vector or path vector routing protocols, so the limitations of the spanning tree algorithms currently used to control VLANs are avoided.

Using ACR to eliminate VLANs will not increase the amount of forwarding state contained in the switches. Today, switches must store a forwarding entry for every MAC address present in the network (otherwise frames are flooded, resulting in terrible performance). The worst possible case for ACR is if every host in a subnet belongs to a different class. This prevents any route aggregation, and forces the routing protocol into flat-address routing – each host needs its own route advertised with its own /32 prefix so that it can list the classes it belongs to. Note, however, that this is no more state than the switches are currently storing for forwarding based on MAC addresses.

We next discuss our implementation of ACR, and evaluate the general framework as well as the implementation.

# 5 Implementation

We have implemented a version of ACR on top of the Click modular router code base [12]. Figure 4 illustrates the four basic entities in our implementation and how they interact: (a) Configuration engine; (b) First-hop router (which also acts the last-hop router for the destination); and (c) Intermediary router. The configuration engine acts as a centralized controller that takes in configuration inputs and sends out configuration information to the individual routers and firewalls. We first describe the configuration setup, network entities, then a version of ACR link-state routing.

## 5.1 Configuration setup and Network entities

The network administrator inputs access control policies at a single location: the Configuration Engine (CE) (Figure 4①). Based on the topological configuration of real world enterprises, a physically-separate control network exists such that the CE can directly connect to all the other entities in the network (routers and firewalls) and specify configuration information for each. Apart from the configuration inputs, the CE needs to be aware of the end-hosts/prefixes that connect to each first-hop and last-hop router. Using the access control rules provided, the CE determines the configuration that is to be distributed to the boundary routers (first and last-hop routers). The current CE implementation assumes that each prefix's subnet connects to a single first-hop router but it can easily be extended to the case where this assumption does not hold. At the first-hop router (Figure 4②), the CE specifies the source classes and at the last-hop router (Figure 4③), classes associated with the destination prefix(es) are modified. The only intermediary routers that are configured by the CE are those that directly connect to a firewall. Any fine-grain access control specification at the CE is installed at the firewall.

Next, we briefly elaborate on the remaining three entities. The code corresponding to the *Intermediary router* implements the simple ACR route computation and forwarding mechanism described earlier. The *First-hop router* uses the same code base as the Intermediary router with two additional functionalities: (a) identification and marking of the class corresponding to each packet; (b) propagating of the set of prefixes that the first-hop router connects to, and the set of classes associated with each prefix. The firewall in our case is similar to a router except with an additional *access filter list*. In our current implementation, the access filter list supports fine-grain access control rules of the form `<srcAddr, dstAddr, dstPort, protocol, accept/deny>` and does not support any deep-packet inspection rules. In our implementation,

only rules that accept packets are used, and the default action is to drop unmatched packets.

### 5.2 Link-State ACR

Next, we describe the propagation of access information by the routing protocol. We implemented a simple version of ACR-based link-state routing where every router maintains the topology of the entire network. Every first-hop router announces a set of prefixes that it connects to, as well as their associated classes (Figure 4④). Here, a prefix represents the smallest granularity on which access control policy is applied. For example, if control is applied on a per-host basis, the first hop router advertises /32 IP addresses. Typical announcements can vary between /24 to /32 prefixes depending on the required granularity within the enterprise. While one can extend the implementation to support aggregation of classes across prefixes, our current implementation does not support it.

**Route computation:** We compute class-based routes on a per-prefix granularity (Figure 4⑤). The route computation process is a straightforward per-class shortest-path computation with class information along the final edges for every path. Hence, the route computation overhead is small.

For t-boxes re-advertising reachability, the intended class transformation (*e.g.* from a prefix-class, say 1.2.3.0/24-C to another prefix-class combination, say 1.2.3.0/24-D) is installed by the CE. Thereafter, the t-box first computes the shortest path cost to 1.2.3.0/24 using class C, and advertises reachability with the same cost, but with class D instead.

**Forwarding process:** At the ingress, or first-hop router, the data packet is tagged with the appropriate class (§4.3), or dropped if there is none (Figure 4⑥). In general, since changes to end-hosts are not necessary, class information is inserted in the form of a shim layer between the link and IP headers. To minimize the additional overhead of classification at the first hop router, we implemented an efficient hash-based process to perform the set intersection operation of classes. While we maintain forwarding entries on a per-class basis, often the amount of state associated with the forwarding table is very small. For the majority of routes not traversing t-boxes, the routing table can be simplified to maintaining a simple prefix-based routing table and a list of allowed classes per prefix. If t-boxes are to be traversed, this implies the possibility of differing next-hops depending on the class, and hence additional routing table entries.

## 6 Evaluation

In this section, we use a combination of qualitative and quantitative measures to demonstrate that ACR is practical and that it indeed simplifies access control configuration in real-world enterprise networks.

**Qualitative analysis:** To really argue that ACR simplifies access control configuration, we need to show that ACR reduces the amount of configuration work that an administrator needs to perform to achieve a certain objective. However, there exists no standard metric to measure configuration complexity. Here, we introduce a simple configuration complexity metric: the configuration complexity of an access control policy event equals the number of rules in different network entities that an administrator needs to manipulate or check for that event. Based on this metric, we consider a variety of basic but commonly occurring scenarios in enterprise environments and measure the configuration complexity for each scenario across three types of networks: ACR, Core Model and Edge Model (as defined earlier in Section 2). We show that across all these scenarios, the configuration complexity for ACR is lower than that of current configuration models (Core and Edge models).

**Quantitative Analysis:** From a quantitative perspective, we consider the access control policies in four large real-world enterprise networks and demonstrate how these policies can be easily translated to configuration using classes in ACR. The real-world enterprise networks that we consider in our analysis are currently operational large enterprise networks with very tight fine-grain access control requirements. As part of the transformation from access control policies to classes, we show that the resulting number of classes required by ACR in each of the enterprise environments is not high. Finally, we perform simple performance benchmarks on various operations of ACR to show that the performance overhead incurred by ACR is not high.

### 6.1 Qualitative Analysis

To perform a qualitative comparison between ACR and the Edge and Core Models for controlling reachabilty, we consider three basic and regularly occurring scenarios in enterprise networks: (a) adding a new entity to the network; (b) communication cessation between two entities previously allowed to communicate; and (c) addition of a new link. For each scenario, we compare the *configuration complexity* of making the change for each of the three models (Core, Edge, ACR). We define configuration complexity to be the number of entities in the system whose configuration an administrator needs to manipulate or validate when updating an access control policy.

We use the following parameters in our qualitative analysis. We assume that there are a total of $n$ entities for which the network is controlling the reachability, with each entity being defined by source prefix, destination prefix, and/or transport-layer port numbers. Therefore, the maximum number of rules is $O(n^2)$. With respect to the Core model, we use $r_c$ to denote the total number of core routers in the network. Note that in the core model, firewalls are co-located with core routers; hence, the number of firewalls is also assumed to be $r_c$. Similarly, we use $r_e$ to denote the number of edge routers in the Edge model.

#### 6.1.1 Addition of New Entity

**Core:** In the core model, adding a new entity (E) to a network like ComNet requires $O(r_c)$ router checks to ensure packets to and from E traverse intended middle-boxes. On the other hand, subnets that can access E will need to know its existence, thus also necessitating $O(r_c)$ configuration.

With regards to the middle-boxes, rules associated with E can be distributed amongst the firewalls, the number of which is expected to be about $O(r_c)$. Even if the network is one-connected, resulting in all complexity being pushed onto a single firewall, there is still the need to verify $O(n)$ rules.

**Edge:** A network structured like UNet with $r_e$ edge routers would require $O(r_e)$ middle-boxes at the periphery. While there is no longer the need to configure routers to enforce routes when adding a new entity, *all* middle-boxes, where $r_e > r_c$ with high probability, will need to be checked to ensure their rules handle the new entity appropriately.

**ACR:** Addition of E requires knowledge of the classes E can access and receive packets from. This information is obtained directly from the high-level access policies, and installed just once at E's first-hop router.

#### 6.1.2 Communication Cessation

In this scenario, two entities previously allowed to communicate with each other is now forbidden to do so.

**Core:** The simplest way of achieving this is to set the appropriate filter to drop packets sent between the two entities. An alternative, or if firewalls are not in use, is to configure the core routers such that reachability information is withdrawn. Therefore, the complexity of the operations remain unchanged, $O(r_c)$ routers and middle-boxes, and $O(n)$ for rules.

**Edge:** As before, the number of places at which rules have to be altered is $O(r_e)$.

**ACR:** For ACR, either a source's (S) permission to access a class (C) of destinations is revoked, or a destination ceases to accept packets of class C. For the former, the change can be effected once the first-hop router of the source is notified. Similarly, for the destination that stops accepting class C packets, changes in the last-hop router is sufficient to ensure that no packets tagged with that class is forwarded across the last hop. This is in spite of the convergence time required, during which a packet may *begin* to be forwarded, but will ultimately be dropped along the way.

#### 6.1.3 Addition of Network Link

**Core:** Adding a new link in the core model may result in unintentional bypassing of middle-boxes. Link weights in the case of OSPF, and route policies for BGP, have to be carefully adjusted to ensure packets take the intended path. Additional middle-boxes may be installed to monitor traffic traversing the new link. In this case, rules may be moved from other boxes resulting in $O(r_c)$ complexity.

**Edge:** Inserting links in the network itself does not require additional configuration, since the routing protocol automatically takes the new link into account. However, additional links to edge subnets may require duplication of the associated middle-box.

**ACR:** Since the introduction of a new link triggers routing updates but does not result in destinations being assigned new classes, this event cannot result in hosts' packets reaching unintended destinations. Thus, no additional configuration is required.

Table 3: Qualitative Comparison Between Existing Models and ACR

| | C.M. | E.M. | ACR |
|---|---|---|---|
| Entity Addition | $O(r_c)$ | $O(r_e)$ | $O(1)$ |
| Communication Cessation | $O(r_c)+O(n)$ | $O(r_e)$ | $O(1)$ |
| Link Addition | $O(r_c)$ | $O(1)$ | $O(1)$ |

Table 3 summarizes the complexity of each operation for current configuration models and ACR. We observe that across these three basic scenarios, the configuration complexity of ACR is significantly better than current configuration methods. While this notion of configuration complexity is not precise, it does provide a way of visualizing how ACR simplifies configuration when compared to the number of additional checks that we need to perform in current systems to achieve access control.

## 6.2 Quantitative: Evaluating ACR in Real-world Enterprises

In this section, we evaluate the effectiveness of ACR in real-world enterprise networks. We consider real-world access control policies used in four large and varied enterprise environments, all very closed networks with very tight access control requirements, and we describe how these access control policies can be transformed to configuration using classes in ACR. Our analysis is aimed at answering the following questions:

1. What do access control policies look like in enterprise networks?
2. How do we translate access control policies to class definitions?
3. How many classes would ACR need to achieve access control in these networks?
4. How would we aggregate hosts/services to define classes in ACR?

### 6.2.1 Description of Enterprise Networks

In our analysis, we consider four different enterprise networks all that belong to the Core Model with tightly constrained access control policies (*i.e.* each source only has access to a limited set of destinations/services):

**ManageNW:** The large commercial entity that we study is a huge amalgam of individual enterprises each dedicated to providing a specific functionality. ManageNW is a large management network that forms the overall management backbone of the entity that interconnects management devices in different enterprises. Within ManageNW, there are several access control policies that restrict administrators within a specific enterprise from accessing devices (routers, firewalls) within this backbone or in other enterprises. ManageNW is a large network that serves nearly 65 individual subnets ranging from /16 to /24 prefixes. Within ManageNW, there are tight restrictions on access control across subnets.

**CorporateNW:** CorporateNW is an internal network consisting of corporate resources such as HR and email systems for various subnets throughout the enterprise. These various subnets consist of multiple /16 prefixes. The access control policies dictate which of the corporate resources can be utilized by groups within the enterprise.

**PerimeterNW:** PerimeterNW is a security infrastructure network that controls access to internal servers from external networks through the use of proxy servers. Within PerimeterNW there are access control policies that dictate the proxy servers that can communicate with internal servers, this results in rules that are /32 based. PerimeterNW has over 250 individual hosts and almost all of the access control policies are specified at the host level.

**AuthNW:** AuthNW forms a large authorization network that entities outside of the commercial enterprise use to access resources and services within the enterprise. Given the large number of outside firms that require access to the enterprise, AuthNW is composed of over 70 prefixes (many of which being /24) and nearly 2000 individual internal hosts (not overlapping with the prefixes). Many of these internal hosts are servers, databases, proxies, firewalls, routers that outside entities connect to. Finally, many of the access control policies in AuthNW are at the level of a host or a group of hosts.

### 6.2.2 Translating Access Control Policies to Classes

There are several possible class assignments that can correspond to the same set of access control policies. We present one such class assignment mechanism which we term *Greedy-aggregation*. First, we identify the basic entities in the system which represents the smallest granularity of prefix (could be a /32 address signifying a host) over which ACL rules are specified in the network.

Once we identify these entities, we model the access control rules in the form of a bipartite graph from source entities (host or prefix) to destination entities (host or prefixes). We then identify aggregate groups of the form $(src - grp, dst - grp)$ where $src - grp$ and $dst - grp$ each represent a group of entities such that any entity within the $src - grp$ can access any in the $dst - grp$. We use a simple greedy algorithm to identify these aggregate groups such that every access control rule is captured in at least one group. While one can define a class optimization problem that attempts to minimize the number of classes for a given access control matrix, a detailed discussion of such a problem is outside the scope of this paper.

### 6.2.3 Applying ACR to Enterprises

Table 4: Number of classes of ACR in real-world enterprises

| Enterprise Network | Number of Entities | Number of classes |
|---|---|---|
| ManageNW | 373 | 105 |
| CorporateNW | 400 | 140 |
| PerimeterNW | 267 | 40 |
| AuthNW | 2110 | 640 |

Table 4 indicates the total number of classes required by ACR for each of the four enterprise networks using the Greedy-Aggregation based class allocation mechanism. We make three observations. First, as per the Greedy-aggregation approach, the number of classes required by ACR is small. The number of classes should be considered relative to the total number of entities; often, the total number of entities may be much smaller than the number of end-hosts given that many entities may represent prefixes. Second, the number of classes corresponding to a single host is very small. In all these networks, this number ranged typically from 1 to 10. Hence, the class-based routing table at every first hop router is small. Later in our performance benchmark study, we show that the overhead of class-based lookup is minimal. Third, one common trend across all networks is the structure of the aggregated classes. In many of the aggregated groups, the $src - grp$ and the $dst - grp$ often referred to entities in the system which are physically not co-located. The implication of this is that in real enterprise networks, the set of entities that have similar access control policies are often dispersed. Hence, it appears that real world enterprises will greatly benefit from the way ACR decouples the access policies of a host from its IP address, allowing a class to represent directly a logical network of disparate hosts that are not physically co-located and do not share a common IP prefix.

In summary, this analysis shows that ACR can easily be adopted in real-world enterprise networks to provide access control, and that the corresponding number of classes invoked within ACR is also small.

### 6.3 Quantitative: Performance Overhead of ACR

In this section, we describe micro-benchmarks that measure the per-packet processing time in ACR in comparison to a normal routing protocol. These benchmarks are performed using our implementation on a Intel Dual-core Pentium 3.40 Ghz processor machine running Click version 1.5.0.

Table 5: ACR forwarding delay as a function of routing table size

| Routing Table Size | ACR ($\mu$sec) | non-ACR ($\mu$sec) |
|---|---|---|
| 10 | 1.58 | 1.19 |
| 25 | 1.60 | 1.25 |
| 50 | 1.62 | 1.29 |
| 100 | 1.72 | 1.34 |
| 250 | 2.23 | 1.61 |
| 500 | 2.97 | 2.11 |
| 1500 | 5.46 | 4.56 |
| 3000 | 9.39 | 7.77 |

Table 5 shows the average time, in microseconds, for the class-based forwarding engine to process a packet for varying routing table sizes for both ACR and normal routing (non-ACR). For ACR, we analyze the performance from the stand-point of a first-hop router which has to perform class-based lookup and forwarding for every packet. For non-ACR, in comparison, we perform a simple routing table lookup operation. For this ACR benchmark, we installed 20 classes for each source and destination prefix in the routing table. A packet was created with a random destination address (within the specified prefixes) with a randomly generated genuine class and passed to the class-based forwarding engine. Based on these results we observe that the overhead of class-based lookup is relatively small but not insignificant (roughly $20 - 30\%$) and this constant overhead (when computed as a fraction) reduces as the routing table size increases. We view this additional overhead as the tradeoff for having the class-based functionality at the routing layer.

Table 6: ACR forwarding delay as a function of number of classes per routing prefix

| Number of Classes | Routing Table Size | Time ($\mu$sec) |
|---|---|---|
| 10 | 1500 | 5.48 |
| 20 | 1500 | 5.59 |
| 30 | 1500 | 5.89 |
| 40 | 1500 | 6.09 |
| 50 | 1500 | 6.35 |

In the previous case, we fixed the number of classes per prefix to be 20 (which is a relatively high number). Table 6 shows the average time, in microseconds, for the class-based forwarding engine to process a packet for varying numbers of classes for each routing table prefix while keeping the routing table size constant. We observe that the additional computational overhead due to having more classes per routing table entry is small. The implementation uses an efficient hash-based process to perform the set intersection operation of classes.

To summarize, we showed that ACR scales well in terms of routing table size as well as number of classes. Furthermore, we found that the number of classes and network entities in large networks in practice can be considered small, and should be handled easily by routers today.

## 7 Discussion

In this section we discuss various miscellaneous issues related to ACR. In particular, we focus on the dynamic installation of classes at first-hop routers, and the minimizing of classes used in an enterprise network.

## 7.1 Location-Independent Class Assignment

In §4.2.3, we discussed how classification rules can be installed at first-hop routers based on inputs to the Configuration Engine. In general, end-users may be mobile and connect to the network at different locations, thus requiring the installation of rules to be dynamic. The bootstrapping process involves obtaining IP addresses *etc.* for communication with other network elements, as well as classification rule(s) installation. We describe these next.

We begin by permitting all hosts access to basic services in the network, such as DHCP. We set a default classification rule, which says that all packets that do not match any other rule will be assigned a minimal-service class $C_{ms}$. The set of services in $C_{ms}$ as usual can be determined by the network administrator, and the corresponding servers must be configured to accept packets tagged $C_{ms}$. Since end-hosts that have been granted permission to send packets of certain classes may still require access to these basic services later (say to renew DHCP leases), they must continue to have the ability to send class $C_{ms}$ packets.

Next, a logically centralized entity in the network has to install the relevant classification rule. In cases where end-hosts are relatively immobile, this role can be taken on by the Configuration Engine. To accommodate mobile users who authenticate themselves, say via Kerberos [14], the two roles (authentication and configuration) can be co-located at a single server. This co-location allows the end-host to contact a single network entity (Kerberos server) for the sole-purpose of authentication, with the latter installing rules at the first-hop router after a successful authentication. Thus, we eliminate the need for changes in the end-host.

In general, since access control at the network level should involve the administrator as well and not solely the user, the end-host should not be able to directly configure the first-hop router. Furthermore, since the end-host identity should be verified before being granted permission to contact corresponding services, we expect both the authentication and configuration entities to be present. Consequently, the authentication servers are the only network entities that need to be extended to accommodate class-assignment that is independent of the hosts' locations.

## 7.2 Class Optimization

In general, having a smaller set of classes results in faster route computation (and hence convergence) and state required. We now describe an algorithm to reduce the number of necessary classes, and begin with the assumption that at least one physical path exists between any two end-hosts. We model the problem using an *Access Graph*, which is a bipartite graph where each network entity[2] is represented by a unique vertex in each partition. Each entity is represented by a node in partition 1 and a node in partition 2. The former represents the entity's ability to send packets and the latter the entity's ability to receive packets. An undirected edge $(u_1, v_2)$ exists if client $u$'s packets are allowed to reach server $v$, that is, the vertex representing $u$ in partition 1 has an edge to $v$ in partition 2. Since an entity can always communicate with itself, the edge $(u_1, u_2)$ always exists for all vertices $u$.

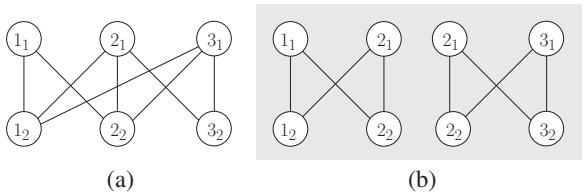


Figure 5: From the bipartite graph in (a), there exists two distinct complete bipartite subgraphs (b), with common vertices $2_1$ and $2_2$ and common edge $(2_1, 2_2)$.

One of our primary focuses is on determining distinct complete bipartite graphs (CBG) in this Access Graph. Unlike convention, edges and vertices may belong in different CBGs. Thus in Figure 5(a), there are two such distinct graphs (Figure 5(b)). Next, the algorithm in Figure 6, where the Access Graph is denoted by $G_{xs}$=$(V_{xs}, E_{xs})$, is used to determine the minimum number of classes required, and returns that number as well as the set of labeled edges.

The intuition behind the algorithm is as follows: as far as possible, we would want to assign the same class to the set of entities that can communicate with one another, *i.e.* those that form a complete bipartite graph. Thus, we begin by finding combinations of such complete graphs (step 1), where we push as many edges into CBGs that are made as large as possible. Next, for each CBG, we assign the same class, which is distinct from those of other CBGs (steps 7-12), this is possible since every entity is allowed to communicate with the rest in the same CBG. Edges that are present in different CBGs can be assigned either class. For the remaining edges, we use a greedy algorithm and assign the same class to edges incident on vertices (say one of which is $v$) with the highest degree (steps 13-27). This means that $v$ is allowed to send or receive packets tagged with the same class. If the number of classes used is less than the previous combination of CBGs, we update the winning combination accordingly (steps 28-30). Applying the algorithm to the access graph in Figure 5, we will end up with three classes: one each for the CBG graphs in Figure 5(b), and another for the edge $(1_2, 3_1)$.

[2] where an entity can be a particular user, group of users, traffic of the same port number, *etc.*.

```
1: find set of combinations of sets of graphs G_cbg s.t. ∀ G_cbg ∈
   G_cbg, where G_cbg consists of CBGs and remaining edges, and
     no_of_CBG_graphs(G_cbg) + no_of_edges_left is minimal
2: set G_win ← (∅, ∅), least_class ← ∞
3: for each set of graphs G_cbg ∈ G_cbg do
4:    set G_tmp ← G_cbg
5:    class_counter ← 0
6:    for each CBG g_cbg ∈ G_tmp,
        where no_of_incident_vertices(g_cbg) > 2 do
7:       find unused class C
8:       for each edge (u, v) ∈ g_cbg and u ≠ v do
9:          set class(u, v) ← C
10:      class_counter ← class_counter + 1
11:   for each vertex u, in descending order of degree (consider-
      ing unlabeled edges) do
12:      if degree(u) = 2 then
13:         find unused class C
14:         for each edge (u, v), u ≠ v do
15:            set class(u, v) ← C
16:         class_counter ← class_counter + 1
17:      else
18:         if in G_tmp,
              ∃ class C' s.t. v_1 = u ∀ class(v_1, v_2) = C' then
19:            ∀ (u, v_2) ∈ g_cbg, set class(u, v_2) ← C'
20:         else
21:            find unused class C
22:            ∀ (u, v_2) ∈ g_cbg, set class(u, v_2) ← C
23:            class_counter ← class_counter + 1
24:   if least_class > class_counter then
25:      least_class ← class_counter
26:      set G_win ← G_tmp
27: return least_class, G_win
```

Figure 6: Pseudo-code for heuristically determining minimum number of classes required.

It is not difficult to see that this algorithm is NP-hard in the general case, so we use a greedy heuristic in obtaining a near optimal assignment. Since the access graph is unlikely to change frequently, we believe it is feasible to compute a near optimal assignment for those networks where it is desirable to minimize the number of classes.

# 8 Related Work

Conceptually, ACR works by creating multiple virtual networks on top of the same physical network, where each class is associated with its own network. Other mechanisms for network virtualization exist, but are not appropriate for access control of the kind that enterprise networks attempt to perform.

VLANs [9] are widely used to aggregate traffic as described in Section 2. However, a host cannot belong to more than one VLAN, whereas ACR supports being a member of multiple classes. VLANs also allow communication between every member of the VLAN, whereas ACR can easily define policies such that members of a class cannot talk to each other.

MPLS [21] is a virtualization technology that assigns labels to packet flows, and sets up corresponding paths through the network. However, using MPLS to implement a class among $n$ hosts would require setting up and maintaining $O(n^2)$ Label Switched Paths (LSPs). Further, MPLS by itself does not solve the problem of making sure the right packets get into the right LSP — that would require still more configuration on the routers. Using MPLS to force traffic through middleboxes might be possible, but again would require an additional technology for directing packets into the right LSP.

BGP-VPNs [22] are similar to MPLS in that they enable the creation of logically separate virtual networks on top of the same physical topology. However, like MPLS, BGP-VPNs are aimed at completely separating traffic. Implementing access control also requires providing conceptually simple and easy to use methods of specifying what reachability should be allowed *between* the virtual networks.

The Multi-Topology OSPF [18] proposal is concerned only with low-level implementation information, and explicitly states that it does not deal with high-level details such as the assignment of classes and their semantics. The proposal is encouraging as it provides an important piece of the overall ACR solution and confirms the fact that virtualization at layer-3 is being seriously looked into.

A different type of solution involves end-host based access control, using protocols such as IPSEC [8] and SOCKS [11]. ACR can work in tandem with these end-host solutions. However, as a network-based solution ACR has the benefits that it can continue to enforce access control policy even in the presence of network-level DoS attacks, end-host misconfigurations, software bugs or other vulnerabilities.

Finally, 4D [6] and SANE [2] both propose centralization of the decision and control planes to provide better security. Rather than explicitly configuring, at a central location, every single aspect of communication, we believe that the same level of immunity from inconsistency can be achieved via distributed routing protocols by introducing the simple concept of a class.

# 9 Conclusion

This paper proposes Access Control Routing that uses classes to define virtual networks, and allows clients and servers to separately decide on the networks they can access or receive packets from. Virtual networks can also be defined to correspond to whether a packet has traversed a middle-box, thus class transformation boxes, by bridging two virtual networks, can force traversal of traffic through it. This separation of configuration, and altering of classes rather than routing or topology, simplifies configuration, which we showed qualitatively. Also, analysis of networks in practice showed that the

number of classes and routes required for full access control is sufficiently small, and can readily be handled by our software-level router implementation.

# References


[1] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. *SIGCOMM Comput. Commun. Rev.*, 34(1):39–44, 2004.

[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 *Proc. 15th USENIX Security Symposium*, Vancouver, BC, Aug. 2006.

[3] W. Cheswick, S. Bellovin, and A. Rubin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley Professional Computing Series, 2003.

[4] Cisco Systems. Spanning tree protocol problems and related design considerations. http://www.cisco.com/warp/public/473/16.html Document ID: 10556, Aug 2005.

[5] T. Dierks and E. Rescorla. RFC 4346: The Transport Layer Security (TLS) Protocol, Apr. 2006.

[6] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang. A clean slate 4D approach to network control and management. *SIGCOMM Comput. Commun. Rev.*, 35(5):41–54, 2005.

[7] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a distributed firewall. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 190–199, New York, NY, USA, 2000. ACM Press.

[8] S. Kent and R. Atkinson. RFC 2401: Security architecture for the Internet Protocol, Nov. 1998.

[9] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks, May 2003.

[10] LAN/MAN Standards Committee of the IEEE Computer Society. IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control, Dec. 2004.

[11] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. RFC 1928: SOCKS Protocol Version 5, Mar. 1996.

[12] R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The click modular router. In *SOSP '99: Proceedings of the seventeenth ACM symposium on Operating systems principles*, pages 217–231, New York, NY, USA, 1999. ACM Press.

[13] J. Moy. RFC 2328: OSPF version 2, Apr. 1998.

[14] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. RFC 4120: The Kerberos Network Authentication Service (V5), July 2005.

[15] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, and B. Tierney. A First Look at Modern Enterprise Traffic. In *Proceedings of Internet Measurement Conference*, October 2005.

[16] V. Paxson. Bro: a system for detecting network intruders in real-time. *Comput. Networks*, 31(23-24):2435–2463, 1999.

[17] Payment Card Industry (PCI) Data Security Standard version 1.1. `https://www.pcisecuritystandards.org/`, Sept. 2006.

[18] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology (MT) Routing in OSPF. `http://tools.ietf.org/wg/ospf/draft-ietf-ospf-mt/`, Nov. 2006.

[19] Y. Rekhter, T. Li, and S. Hares. RFC 4271: A Border Gateway Protocol 4 (BGP-4), Jan. 2006.

[20] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). `http://www.ietf.org/rfc/rfc2865.txt`, June 2000.

[21] E. Rosen, A. Viswanathan, and R. Callon. RFC 3031: Multiprotocol Label Switching Architecture, Jan. 2001.

[22] E. Rosen and Y.Rekhter. BGP/MPLS VPNs. RFC 2547, March 1999.

[23] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: vulnerability-driven network filters for preventing known vulnerability exploits. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 193–204, New York, NY, USA, 2004. ACM Press.

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