Enforcement of Data-Plane Policies in Next-Generation Networks

Shashank Shanbhag and Tilman Wolf
Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA
{sshanbha,wolf}@ecs.umass.edu

***Abstract*—Modern networks not only forward traffic, but also perform a variety of processing operations on packets (e.g., content inspection, transcoding, QoS scheduling). Such data-plane operations cannot be easily coordinated in the current Internet architectures since there is no explicit policy support for packet processing services. As more diverse systems and protocols are deployed in the next-generation Internet, this problem becomes increasingly challenging. In our work, we propose a novel policy enforcement system for data-path functions in the next-generation Internet. Using a formalism to represent policies and automated planning tools, connection request can be adapted to meet the policy requirement of the domains they traverse. We present the theoretical foundations of this approach as well as a prototype implementation based on our network service architecture. Our results show that this approach is an effective solution to enforcing policies relating to the date plane of networks.**

***Index Terms*—next-generation Internet, network service, packet processing, automated planning, data-plane policies.**

I. Introduction

Policies (in the broadest sense) can be found throughout the existing Internet: routing policies are used to determine how to forward traffic and which routes to advertise to neighboring networks (e.g., using border gateway protocol), security policies are used to determine what traffic is admissible in a network (e.g., using firewalls or intrusion detection systems), service-level agreements (SLAs) are used to define quantitative metrics for network performance, etc. These policies can target any layer in the protocol stack from as low as the link layer (e.g., SLA on available bandwidth on link or maximum bit error rate) to as high as the application layer (e.g., firewall rule on permissible application layer protocols). They can also focus on the control plane of the network (e.g., routing policies) as well as on the data plane of the network (e.g., IDS rules on permissible packet payloads).

The current Internet does not have any general support for data plane policies. However, there are numerous systems in the Internet that enforce local data plane policies. Examples include firewalls, intrusion detection systems, etc. These "middle-boxes" can cause significant problems when setting up connections. For example, a firewall may block traffic that is directed toward a particular end-system. Since this blocking happens silently without explicit notification, the transmitting end-system cannot determine why a connection request is unsuccessful. Similar problems appear with the use of intrusion detection system to drop malicious traffic, selective connection resets to throttle peer-to-peer traffic, etc. This implicit handling of policies in the current Internet does not provide any mechanism for communicating entities to negotiate or adapt. Thus, connections may succeed if they happen to meet all policy requirements, but fail in all other cases. This all-or-nothing approach presents a considerable problem in the Internet since it requires careful crafting of policies to avoid potentially undesirable connection denials.

Next-generation Internet architectures proposes the use of network virtualization to diversify systems and communication protocols allowing multiple logical networks with different protocol stacks to share a single network infrastructure [1]. Together with the broad deployment of novel communication paradigms (e.g., content delivery, delay-tolerant networking, sensor networks, etc.) and the availability of custom packet processing functions in routers, the functionality of the data plane is expected to be significantly more complex than in current networks. Therefore, the number of policies determining network operation is expected to increase as well leading to a point where the current approach of manually adapting connection requests to meet local policies of all traversed networks simply no longer scales. Instead, an automated approach to adapting connection requests to meet policies is necessary. The design of such an automated system is the topic of our paper. We focus on policies relating to data plane functionality of the network (i.e., policies that affect how the network handles packets as they are being forwarded). The policies we consider affect how traffic is handled on routers (e.g., "are packets forwarded or dropped?," "is intrusion detection performed on packet payloads?," or "should a packet processing service be carried out on traffic belonging to a certain connection?"). Note that policies relating to the control plane (e.g., routing policies) are a separate issue and not further addressed.

In this paper, we define a formal representation of data plane functionality and policies. By translating this representation of policies from different administrative domains into planning rules, our system can determine how to set up a connection request such that all policies are met (if a solution exists). The system considers arbitrary networking functions, which we call "services," as part of the connection setup. Each autonomous system in the network can specify the properties of acceptable traffic and which services need to be performed on traffic as their local policies. Using an automated planning tool, our system can then determine which services are necessary to

complete a valid connection request. Based on a prototype implementation, we can demonstrate how our system can successfully enforce and adapt policies on such connection requests.

The remainder of the paper is organized as follows. Section II discusses related work. Section III introduces the formal representation of policies that allows us to use automated tools for planning and proposes an automated policy enforcement system. Prototype implementation and evaluation results are presented in Section IV. Section V summarizes this paper.

II. Related Work

Policies in the context of next-generation network architectures have focussed on routing (i.e., the control plane) rather than packet forwarding functionality (i.e., the data plane) [2]. However, there are several network designs that consider per-connection customization of network functionality [3]. For such networks, it is essential to provide automated policy enforcement mechanisms focusing on the data-plane functionality.

Much research has been done in the area of policy-based network management [4]. Earlier work focuses on condition-action rules to support a static configuration based solution, which required human intervention for system and policy reconfiguration. Lymberopoulos et al. [5] propose an automated policy deployment and adaptation framework that permits dynamic configuration of policy parameters and objects in response to changes within the managed environment. The QoS network management framework developed by the IETF Policy Working Group using the X.500 directory schema encodes policies as If-Then conditions. Yoshihara et al. [6] propose a policy parameter adaptation framework that uses a management script expressed in the IETF Policy Working Group's representation targeting differentiated services. Badr et al. [7] propose an autonomic policy (internal and external) based control service that allows runtime system diagnosis, repair and reconfigurations that helps correct the system in events of conflicts or failures with minimum human intervention. The control service uses an extension to the Beliefs, Desires and Intension (EBDI) [8] model to generate new repair strategies.

These policy-based network management frameworks emphasize adaptive policy deployment and parameter reconfiguration focusing on the control plane but do not take into account services applied to the connection. In contrast, our system does not modify existing polices but adapts to existing ones by rearranging the sequence of services or adding new ones, thus respecting local and global policies.

III. Problem Formulation and Proposed Solution

A. Notation

Let P be the set of all possible policies in all autonomous systems. A is the set of all data-plane processing functions (services) in the network. We assume that all the services are standardized and each domain has the ability to implement each of those services. The concept of data-plane services as network functions was addressed in our previous work [9]. Let Δ be the set of all possible states for a connection with $|\Delta| = n$. Any connection c can then be uniquely identified by the tuple, $\langle I_{src}, I_{dst}, \Delta_{src}, \Delta_{dst}, \Delta_{curr} \rangle$ where I_{src} and I_{dst} are the source and destination specifiers respectively. In terms of the current Internet, these will consist of the source and destination IP addresses and port numbers. Similarly, Δ_{src} and Δ_{dst} are the states of the connection c at the source and destination respectively. Δ_{curr} is the current state of the connection at any point during its traversal in the network, $\Delta_{curr} \in \Delta$. Δ_{curr} keeps track of the state changes in the connection as the data is processed by data-plane services along the path.

The state of a connection can be changed by: Local Domain Policies that are state specific or by Data-plane services applied according to global connection requirements. Local domain policies can be represented by the state transition equation: $\delta' = f(P_i, \delta)$ where P_i is the i^{th} policy that changes the state of the connection c from δ to δ' where $\delta, \delta' \in \Delta$. Therefore, a policy is a function of the state of the connection and is essentially a mapping from the state of the connection to the action(s) that has to be taken on that connection in that state. For data plane policies, the actions are the services that have to be applied to the connection. Note that the policies may not affect the state at all, i.e., $\delta = \delta'$. For simplicity, we assume that there is a one-to-one mapping between policies and the states they are applicable to and vice-versa. The cases where multiple policies match a given state or a single policy matching multiple states are not considered.

Similarly, a data-plane service when applied to a connection changes its state according to the transition equation: $\Delta_{dst} = g(\mathring{A}, \Delta_{src})$. $\mathring{A} \in A$ converts state from Δ_{src} to Δ_{dst}. An end-system can request data-plane services to be applied to its connection by a connection request defined by the tuple $c = \langle I_{src}, I_{dst}, \Delta_{src}, \Delta_{dst}, \Delta_{curr} \rangle$. The network is responsible for deriving the sequence of services that achieve this change. In our previous work [9], we solve this "service composition problem" by reducing it to a planning problem as follows: If $A = \{A_1, A_2, ..A_a\}$ is the available set of network services, and c is the communication request given represented by the tuple $\langle I_{src}, I_{dst}, \Delta_{src}, \Delta_{dst}, \Delta_{curr} \rangle$, find an ordered sequence of services, $(A_{x_1} \rightarrow A_{x_2} \rightarrow ... \rightarrow A_{x_k})$ such that $A_{x_i} \in A$ for $1 \leq i \leq a$ and $c \models (A_{x_1} \rightarrow A_{x_2} \rightarrow ... \rightarrow A_{x_k})$. The symbol $\rightarrow$ determines the sequence in which the services have to be performed. This problem is solved by reducing it to a "planning problem," where the services are equivalent to actions with each service having its own precondition and an effect on the current state Δ_{curr}. The entire composed sequence of services that meet the connection requirements is called a "plan" denoted by π.

From the two transition equations we see that both policies and services can be specified in terms of states in which they are applicable. This allows us to merge both data-plane policy rules and service specifications into one singular database.

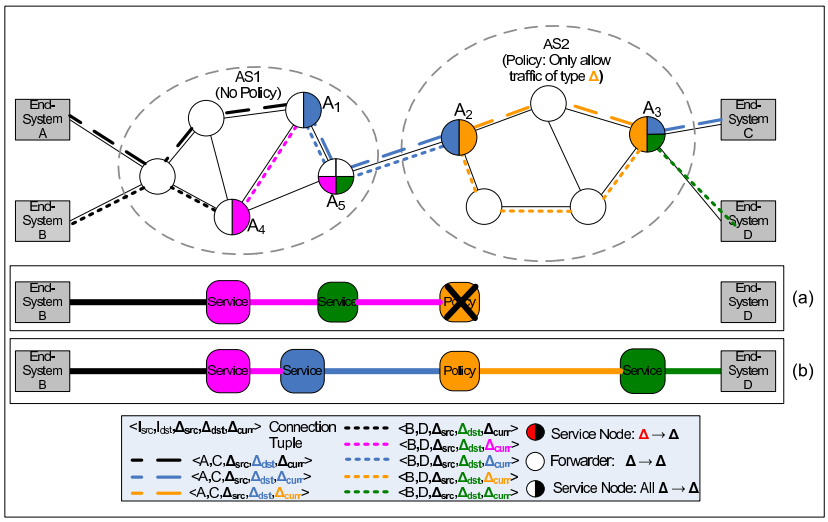

Fig. 1. In the absence of policies, the network seamlessly takes care of the request between end-systems A-C by adding the service A_1 in the data path. Similarly, the B-D request results in the $A_4 \rightarrow A_5$ composition. However, AS2 has the policy of only letting through traffic with the state "orange" while AS1 has no local policies. If the network ignores the AS2 policy and composes the sequence, packets belonging to the connection will be dropped at A_2 even though the connection request was satisfied.

B. Service Recomposition Problem Formulation

The importance of considering data-plane policies while composing a sequence of services is illustrated in the two Scenarios (a) and (b) in Figure 1. Data-plane services have global scope and are applied to the entire connection, i.e., it does not matter where the services are placed along the connection as long as the order of the services is correct. On the contrary, certain policies have local scope and have to be enforced locally. In such cases, state agnostic policy enforcement can lead to disastrous results for the connection as illustrated in Scenario (a). Scenario (b) illustrates the case where data-plane policies along the path have been taken into account while planning the sequence. The result is the addition of two complementary services at A_2(blue to orange) and A_3(orange to blue and green), A_1(all data to blue) because A_2 only processes 'blue' traffic according to policy, and A_5(pink to green) has been replaced by A_3 and placed at a different node in the network to accommodate AS2 policy.

We are interested in the case where an initial service sequence has been composed to satisfy a connection request but the context in which it needs to be executed has changed from the expected context because of data-plane policies on the path. If π_k be a plan with k services $(A_1, A_2, ..., A_k)$, composed for the connection request $c = \langle I_{src}, I_{dst}, \Delta_{src}, \Delta_{dst}, \Delta_{curr} \rangle$. If a data-plane policy is enforced at some step $x (1 \leq x \leq k)$ in the plan π_k resulting in an incompatible state Δ_{x+1} at x as input to the service A_{x+1}. This is defined as the "service recomposition problem" and can be solved by either planning from scratch (which is redundant because changing the sequence of services before x is unnecessary) or by reducing it to a "plan repair problem" as follows: Let the planning problem be described by the following tuple, $\langle \Delta_{src}, \Delta_{dst}, \pi \rangle$. where $\pi = (A_1, A_2, ..., A_k)$. Following enforcement of a data-plane policy at a step x in the plan π resulting in the deviation from the expected state at step $x+1$, the new planning problem is described by the tuple, $\langle \Delta_{x+1}, \Delta_{dst}, \pi' \rangle$. Then, the repaired plan is achieved by combining $\pi' = (A'_1, A'_2, ..., A'_j)$ with $\pi = (A_1, A_2, ..., A_x)$.

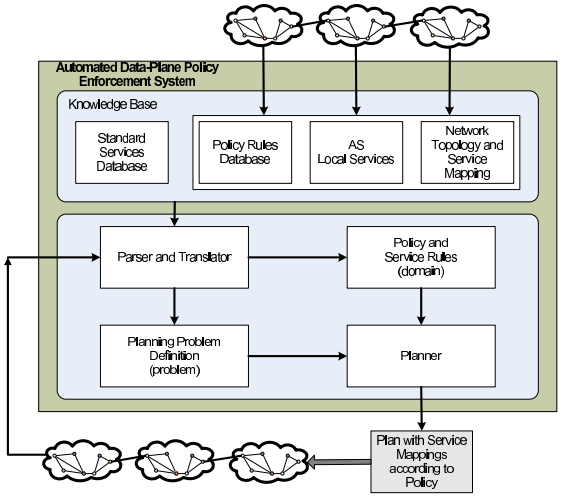

Fig. 2. Automated Data-plane Policy Enforcement System.

C. Automated Data-plane Policy Enforcement System

The policy enforcement system (Figure 2) has three major components: the knowledge base, the parser/translator and the planner. The knowledge base consists of the standardized service library with service specifications, the policy rules database that stores all policies along with their mapping to various autonomous systems in case of local policies, non-standard services specifications that may be implemented locally in an AS and finally, the topology information of all ASs. The parser/translator parses the service specifications and the policy rules database and converts them to a planning language the planner can operate on and merges them into a singular database (planning domain). It also parses connection requests

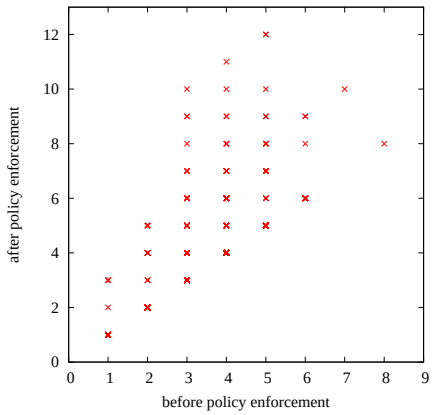

Fig. 3. The number of services before and after policy enforcement. Services are either added or removed after policy enforcement as the planner repairs the service composition.

from end-systems to create a planning problem definition for the planner to solve. Given the problem, the planner then tries to find a valid plan that meets the connection requirements. In cases where enforcement of policies results in a change in the expected final state, the planner repairs the plan in order to incorporate the data-plane policy. The plan with the service mappings is then relayed to the autonomous systems.

IV. Prototype Implementation and Results

Automated mechanisms rely heavily on the state of a process. For the purpose of representing state, we use the "semantic tree" representation proposed in our earlier work [9]. The tree structure includes class hierarchies inherent in data and communication characteristics and is a general and comprehensive representation of network traffic state. Using this representation, a wide range of policies and services can be specified. To this end, we use the W3C Web Ontology Language (OWL) to describe semantics and relay state to the planner. The planner used is LPG [10][1]. The simulation[2] was run on a 64-bit Quad-Core Intel Xeon 5300 with 2GB memory with 2x4MB shared cache. For 2000 simultaneous connections, the system takes about seven seconds to repair the plan averaging about 3.618ms per connection. This is primarily due to a centralized implementation where the planner has to deal with policies, services and topology information of the entire network. A more decentralized approach may result in a decrease in repair times.

Figure 3 compares the lengths of the service sequence before and after policy enforcement. If the enforcement of policies on the connection by the ASs results in a wrong sequence, the sequence is repaired by the system by rearranging the services in the sequence, adding new services or removing existing ones to meet the connection requirements while enforcing policy at the same time.

[1]LPG is a fully automated domain-independent planner based on local search and planning graphs. It operates on domains and problems described in PDDL (Planning Domain Definition Language) which is an action-centric description language planning problems.

[2]96 nodes were organized into 12 ASs with each AS containing eight nodes. Traffic state was defined by six dimensions chosen from the semantic tree. The service nodes are capable of performing a maximum of two services. Each AS enforced a maximum of two policies every connection and each policy was generated in a random manner. The policies mostly consisted of forcibly adding a service to the composed sequence so as to alter the plan.

V. Conclusions

Our work explores the issue of enforcement of policies on data-plane operations in the network. The increasing number of packet processing functions along with the need to enforce policies in the data-plane presents a unique set of problems that may lead to the failure of the connection. Thus, it is important to determine the process to adapt connection requests to meet these policies. Our system uses a formal representation of packet processing functions and policies relating to them in order to automate the process of policy enforcement. Using a planning tool, connection requests can be re-composed to adjust the connection semantics to meet local policies. Our prototype implementation shows that this approach can adjust connection requests appropriately and thus can enforce data-plane policies. We believe that this work presents an important step towards a comprehensive architecture that allows explicit representation, exchange, and enforcement of data plane policies in next-generation networks.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0626690.

References

[1] J. S. Turner and D. E. Taylor, "Diversifying the Internet," in *Proc. of IEEE Global Communications Conference (GLOBECOM)*, vol. 2, Saint Louis, MO, Nov. 2005.

[2] D. Clark, K. Sollins, J. Wroclawski, D. Katabi, J. Kulik, X. Yang, B. Braden, T. Faber, A. Falk, V. Pingali, M. Handley, and N. Chiappa, "New Arch: future generation Internet architecture," Defense Advanced Research Project Agency, Tech. Rep., Dec. 2003.

[3] T. Wolf, "Service-centric end-to-end abstractions in next-generation networks," in *Proc. of Fifteenth IEEE International Conference on Computer Communications and Networks (ICCCN)*, Arlington, VA, Oct. 2006, pp. 79–86.

[4] J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation (The Morgan Kaufmann Series in Networking)*. Morgan Kaufmann Publishers Inc., 2003.

[5] L. Lymberopoulos, E. Lupu, and M. Sloman, "An adaptive policy based management framework for differentiated services networks," in *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, Washington, DC, USA, 2002.

[6] K. Yoshihara, M. Isomura, and H. Horiuchi, "Distributed policy-based management enabling policy adaptation on monitoring using active network technology," in *12th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Nancy, France, 2001.

[7] N. Badr, A. Taleb-Bendiab, and D. Reilly, "Policy-based autonomic control service."

[8] N. Badr, "An investigation into autonomic middleware control service to support distributed self-adaptive software."

[9] S. Shanbhag, X. Huang, S. Proddatoori, and T. Wolf, "Automated service composition in next-generation networks," in *Proc. of The International Workshop on Next Generation Network Architecture (NGNA)*, Montreal, Canada, 2009.

[10] A. Gerevini and I. Serina, "Lpg: A planner based on local search for planning graphs with action costs," in *Proc. of the Sixth International Conference on Artificial Intelligence Planning Systems (AIPS)*, Toulouse, France, 2002.