Automated Composition of Data-Path Functionality in the Future Internet

Shashank Shanbhag and Tilman Wolf, University of Massachusetts

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-path functionality needs to be composed suitably to ensure correct operations and adherence to policies put in place by different entities in the Internet. Currently, there exists no explicit support for describing the semantics of packet processing services, for composing these services, and for ensuring that policies are considered. In our work, we propose a novel system for representing data-path functionality and policies in such a way that per-connection configurations can be composed automatically. 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 toward scalable and autonomic handling of data-path functionality in the future Internet. Index Terms next-generation Internet, network service, packet processing semantics, automated planning, data-path policies.

The Internet has been very successful in providing data communication connectivity between end-systems. As the Internet has matured, additional functional demands have required that routers implement packet processing operation beyond simple packet forwarding. Examples of this type of data-path functionality are firewalling, network address translation, intrusion detection, content caching, etc. Recent research efforts to redesign the network architecture of the future Internet aim to further extend this data-path functionality to include protocol processing operations that are dynamically deployable [1].

The availability of different features in the data path of networks leads to numerous choices about which functionality to instantiate when establishing a connection. While these choices provide a powerful tool for implementing novel communication paradigms and protocols, they also present a major challenge: How can the functions for a connection be composed in such a way that requirements are met and the network operates efficiently? In this context, it is necessary to consider the requirements of the application that uses the connection, the functionality that can be provided by data-path functions in the network, and the administrative policies that are put in place by network operators (or other entities). Requirements may include functionality (e.g., certain type of protocol operation) and performance (e.g., certain amount of available bandwidth). Since performance requirements have been studied extensively in the context of Quality of Service (QoS) research, we focus our work on managing data-path functionality. An example of a composition problem in this context is: "Send video from source *A* to receiver *B* in format *X* and to receiver *C* in format *Y* while considering that network *N* blocks certain formats and requires that all traffic be inspected."

The sheer scale of the Internet makes it clear that the management of functionality for this kind of network cannot be handled "manually," and instead, there is a need for *automated management*. Our work presents a step in this direction.

We present a novel system for automated composition and policy enforcement of data-path functionality in next-generation networks. The key aspects of this system (and how they differ from current approaches) are:

- Per-connection composition and policy enforcement: Our system handles each connection independently and thus permits fine-grain composition and policy enforcement (instead of applying general composition patterns and policies to all traffic).
- Automated adaptation: The system is entirely automated and thus can enforce policies by adapting connection requests without the involvement of administrators (instead of requiring manual configuration of firewalls or other network systems).
- Runtime operation: The automated composition and enforcement component permits the system to determine constraints at runtime (rather than pre-computing all possible rules and statically enforcing them). Thus, the system can quickly adapt to changes in available data-path functionality and policies during network operation.

The key aspect of our system is the use of a formal representation of the semantics of a data connection and the functions that can operate on it, which we call "services." Using this formalism, policies from independent administrative domains can be expressed. By translating these semantics and policies into planning rules, our system can determine how to set up a connection request such that all requirements are met (if a solution exists). Each autonomous system in the network can specify the properties of acceptable traffic and the services

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

0890-8044/11/$25.00 © 2011 IEEE

that need to be performed on traffic as their local policies. Using a planning tool, our system can then determine the services necessary to complete a valid connection request. Based on a prototype implementation, we can demonstrate how our system can compose data-path functionality and successfully enforce and adapt to policies on such connection requests.

Automated Composition of Data-Path Functionality

In this section, we will briefly discuss the background on data-path services and network policies before presenting the formalism that enables us to automate composition and policy enforcement.

Services in Network Data Path

A network service is any type of processing function that is performed in the data path of the network such as packet forwarding, network address translation, multicast, QoS routing, privacy, etc. These services are applied to a subset of (or all) packets in the network. We focus on decomposing the processing functionality offered by all layers of the architecture into basic components. We expect that this set of services is standardized (e.g., through IETF) and the number of services is in the order of tens. The end-system applications can then customize their connections using a customized sequence of services that provides the necessary flexibility.

Policies

Policies (in the broadest sense) can be found throughout the existing Internet: routing policies, which are used to determine how to forward traffic and the routes to advertise to neighboring networks (e.g., using border gateway protocol (BGP)); security policies (e.g., using firewalls or intrusion detection systems (IDS)); service-level agreements (SLAs), which are used to define quantitative metrics for network performance, etc. These policies can target any layer in the protocol stack (e.g., SLA on available bandwidth on link or maximum bit error rate).

We focus on policies relating to the data path functionality of the network, i.e., policies that affect how the network handles packets as they are being forwarded. For example, "are packets forwarded or dropped?" 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 addressed. Numerous systems in the Internet enforce local data path policies (e.g., firewalls, IDS, etc.). This can cause significant problems when setting up connections. A firewall may block traffic that is directed toward a particular end-system. 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. 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 article.

Representation of Data-Path Services and Policies

Next-generation networks are expected to deploy a large number of custom processing functionality in the data path. This also results in an increase in the number of policies controlling network operation. Manual configuration of per connection policies no longer scales. Therefore, there is a need to automate this process. To achieve this, a formal representation of policies and services is necessary.

Flows and States

The automation of service composition and data-path policy enforcement depends on the notion of network traffic state. Data-path policies are applied to network traffic, and services process traffic depending on the whether the traffic belonging to a connection satisfies the state and flow requirements of the policy and/or service.

In the current Internet, flows help distinguish between packets belonging to different connections. A flow is a host-to-host communication path identified uniquely by the five-tuple: source and destination addresses, port numbers and the transport protocol. This allows us to distinguish between packets from different flows and apply different policies depending on the priority of flow (e.g., dedicated bandwidth, low latency and controlled jitter for real-time traffic, best effort service for nonreal-time traffic, etc.). Although the five-tuple can be used to provide control plane policing mechanisms, it is inadequate for data-path policing, especially in next-generation Internet architectures where the emphasis is on services in the data path. A common example is the failure when a compression policy is enforced on encrypted traffic based only on the five-tuple. Therefore, next-generation services will depend heavily on the semantics of the data being transferred in addition to the five-tuple. Since data-path policies are centered on services themselves, policy definitions must also include semantics of the data.

Application of data-path services to data traffic also changes the semantics (state) of the data, rendering the network traffic incompatible with the services that may be applied later in the path. Therefore, keeping track of changes in semantics enables the rearrangement of services so that the connection is successful. The network traffic can be modeled as progressing through a series of states as various services process the data along the path. Therefore, the state of the connection at any point along the path is defined by the semantics and the five-tuple of the traffic flowing in the connection.

Notation

We use the notation in Table 1 to formalize representation of data-path policies and formulate the problem statement. The traffic in the connection traversing a network can have its state changed by either data-path services or local domain policies.

Data-Path Services — End-systems can request processing in the data path in two ways:

- As a sequence of services to be applied to the connection.
- In terms of the states of the connection at the source and destination, through a connection request, c. The network is responsible for deriving a sequence of services that achieves this. Services are fully described using "preconditions" (logical conditions that need to be satisfied in order for the service to be executed) and "postconditions" (the transformations in the state caused by the application of the service). Consider a simple connection request that takes a single service $\mathring{A} \in A$ to convert state Δ_{src} to Δ_{dst}. Then, the relation between the service $\mathring{A}$ and the states can be represented by the following state transition equation:

$$\Delta_{dst} = g(\mathring{A}, \Delta_{src}) \quad (1)$$

Local Domain Policies — Let $P_{i,j}$, be the ith local policy in the domain D_j that changes the state of the connection c from δ to δ' where $\delta, \delta' \in \Delta$. This can be represented by the state transition equation:

$$\delta' = f(P_{i,j}, \delta) \quad (2)$$

Therefore, a policy is a function of the state of the connection and a mapping from the state of the connection to the action(s) that has to be taken on that connection. For data path 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.

From the state transition Eqs. 1 and 2, we see that both polices and services can be specified in terms of states in which they are applicable, allowing us to aggregate both data-path policies and services.

Data-Path Services: Automated Service Composition Problem

We state the service composition problem as follows: If $A = \{A_1, A_2, \ldots A_a\}$ is the available set of network services, and c is the communication request given represented by the tuple $c = \langle I_{src}, I_{dst}, \Delta_{src}, \Delta_{dst}, \Delta_{curr}\rangle$, find an ordered sequence of services, $(A_{x1} \rightarrow A_{x2} \rightarrow \ldots \rightarrow A_{xk})$ such that $A_{xi} \in A$ for $1 \leq i \leq a$. The symbol $\rightarrow$ determines the sequence in which the services have to be performed, i.e., $x \rightarrow y$ implies the service x should be performed before service y. 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."

Applying Data-Path Policies: Service Recomposition Problem

Using state information helps us merge policy rules and service specifications into one database. This makes service placement non-trivial because services have global scope (applied to the entire connection) while certain policies have local scope (enforced within a domain). Furthermore, state agnostic enforcement can lead to disastrous results for the connection itself. Therefore, the composed sequence of service itself needs to be repaired to take into account the local domain policies. This problem is illustrated in Fig. 1. The connection fails because the local policy forces a service whose precondition does not match the current state of the connection. The example illustrates the importance of taking into account the state of the connection while implementing data-path policies. This is called the "service recomposition problem" and is defined as follows:

Given π_k and Δ_{src}, the entire sequence of states $(\Delta_1, \Delta_2, D_3, \ldots, \Delta_{k+1})$ can be derived using Eq. 1. Here, $\Delta_1 = \Delta_{src}$ and $\Delta_{k+1} = \Delta_{dst}$. Let a data-path policy be 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}, thus rendering the sequence meaningless. This modified plan is repaired by either rearranging the services in π_k or adding services to π_k. This is called the "plan repair problem" and is solved as follows: Let the planning problem be described by the following tuple, $\langle \Delta_{src}, \Delta_{dst}, \pi\rangle$ where Δ_{src}, Δ_{dst} are the initial state and target state, respectively, and π is the service sequence (plan) that achieves the target state from the initial state in k steps $(A_1, A_2, \ldots, A_k)$. Then, following a change in the plan π at any ß step x resulting in 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 $\pi' = (A'_1, A'_2, \ldots, A'_j)$ with $\pi = (A_1, A_2, \ldots, A_x)$.

D	Set of all autonomous systems (domains), $\lvert D\rvert = d$
P	Set of all policies in all domains
A	Set of all services in the network, $\lvert A\rvert = a$
Δ	Set of all possible states of a connection, $\lvert\Delta\rvert = n$
I_{src}	Source address (e.g., source IP/port number)
I_{dst}	Destination address (e.g., destination IP/port number)
Δ_{src}	Connection state at the source
Δ_{dst}	Connection state at the destination
Δ_{curr}	State of the connection at any point in the network
c	The connection c uniquely identified by the tuple, $\langle I_{src}, I_{dst}, \Delta_{src}, \Delta_{dst}, \Delta_{curr}\rangle$.
π_k	A sequence of k services, denoted by $(A1, A2, \ldots, Ak)$ for the connection request denoted by c.

Table 1. *Notation.*

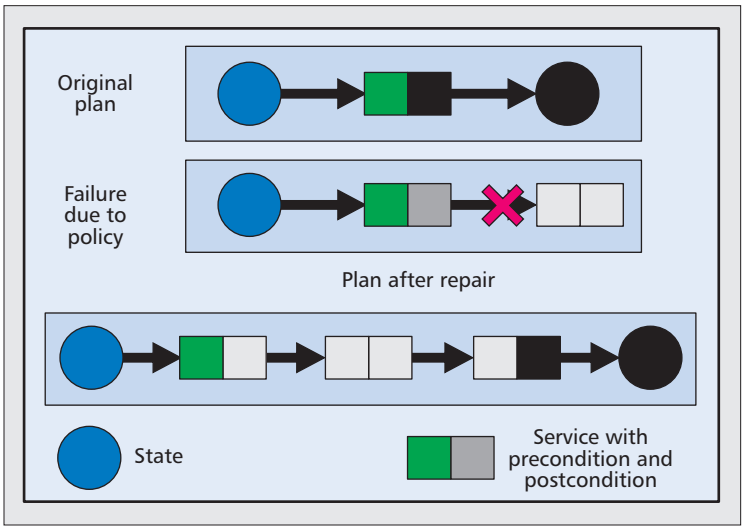

Figure 1. *State agnostic policy enforcement leads to a failure of the connection. A plan repair is initiated by adding services or rearranging the plan around the enforced policy such that the connection is successful.*

Prototype Using Network Service Architecture

Semantic Tree

One issue that occurs in the design of the automated policy enforcement system is: Which characteristics are important enough to be examined when describing the state a policy or a service is mapped to? After an extensive study of the services, applications, and communication paradigms in the current Internet, we designed a semantic tree structure (Fig. 2) to represent state in terms of data and communication characteristics. The tree structure includes class

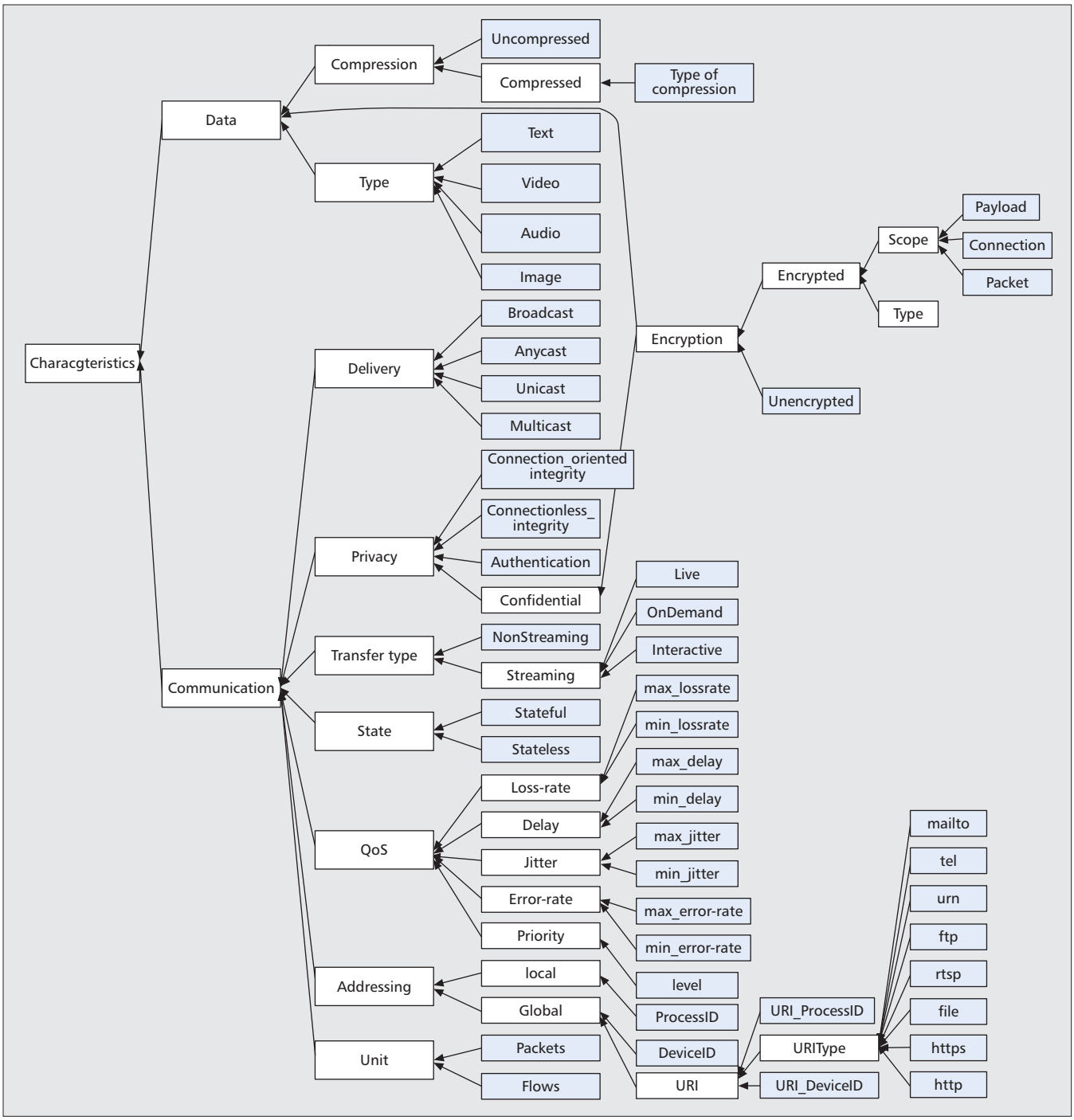

Figure 2. *Semantics for representing characteristics.*

hierarchies inherent in data and communication characteristics and forms a comprehensive method of representing state. Using this representation, a wide range of policies can be defined and implemented. Therefore, the semantic tree structure is a general and comprehensive representation of state with respect to network traffic.

Automated Service Composition and Data-Path Policy Enforcement System

The automated system (Fig. 3a) has three major parts: the knowledge base, the parser/translator, and the planner. The knowledge base consists of the standardized service library with service specifications (preconditions and postconditions), the policy rules database that stores all policies along with their mapping to various autonomous systems in case of local policies. The knowledge base also contains specifications for non-standard services that may be implemented locally in an autonomous system and finally, the topology information of all the autonomous systems. The parser/translator parses the service specifications and the policy database into planning rules. The parser/translator also parses connection requests from end-systems to create a planning problem definition. The planner then tries to find a valid plan that meets the connection requirements and incorporates data-path policies.

The whole system proceeds in steps as follows (Fig. 3):

- All autonomous systems add their data-path policies, information about the local services and the AS topology to the

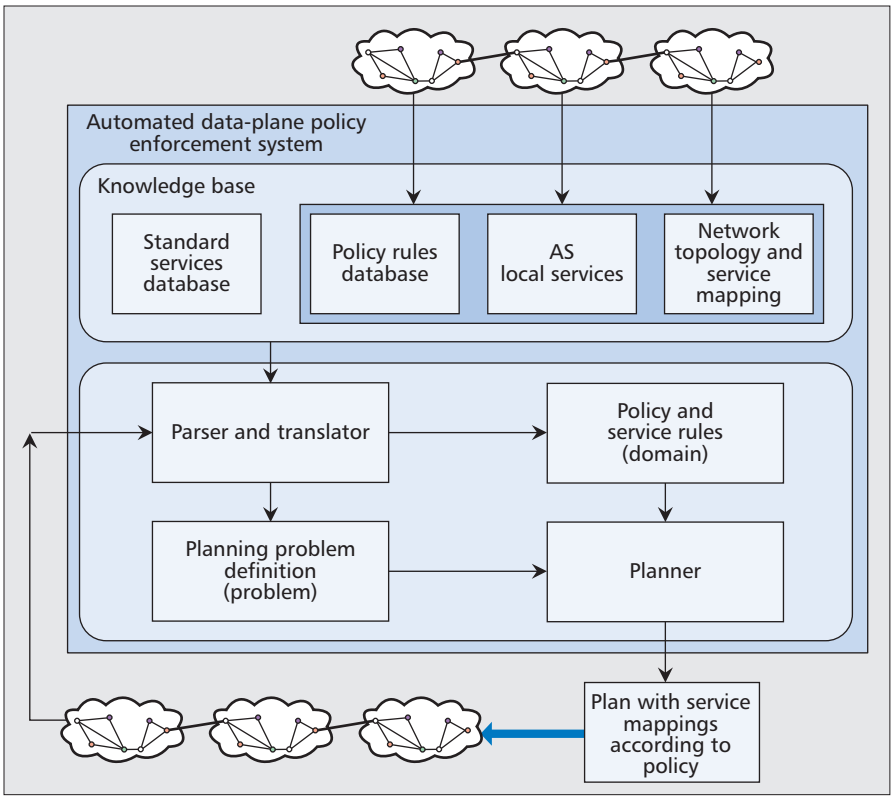

Figure 3. *Policy enforcement system.*

knowledge base. The parser/translator parses the policy rules, service specifications and network topology information and translates them into a language specific to the planner being used.

- An end-system (Fig. 4) trying to initiate a connections sends the request to the system in the form of the tuple.
- The parser/translator parses the request and creates a problem definition. The planner takes the domain and problem definition as input and tries to find a plan that satisfies the connection requirements.
- In cases where policies are enforced, the planner takes care to replan and/or repair the service composition in order to incorporate the data-path policy in the final plan.
- The plan with the service mappings is then relayed to the autonomous systems.

Prototype Implementation

The first step in implementing the proposed automated system is to represent state in terms of semantics of network traffic (Fig. 2). A number of semantic markup languages exist for describing the properties and capabilities of web services as well as communication level descriptions of messages and protocols necessary for the operation of these web services. We use the W3C Web Ontology Language (OWL)[1] for this purpose. The planner used is LPG [2] (Local search for Planning Graphs), a fully automated domain-independent planner. LPG uses the PDDL (Planning Domain Definition Language) for describing domains and problems. A planning problem is described using a domain description that includes the actions (behavior of the system), and a problem description (goals to be reached). The parser and translator in Fig. 4 parse and convert OWL definitions of service and policy definitions to domain specification for LPG to operate on.

Evaluation Results

Automated Service Composition

We built a prototype described earlier on a 64-bit Quad-Core Intel Xeon 5300 with 2 GB memory with 2 × 4 MB shared cache. Table 2 describes the services implemented along with

[1] *http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.2*

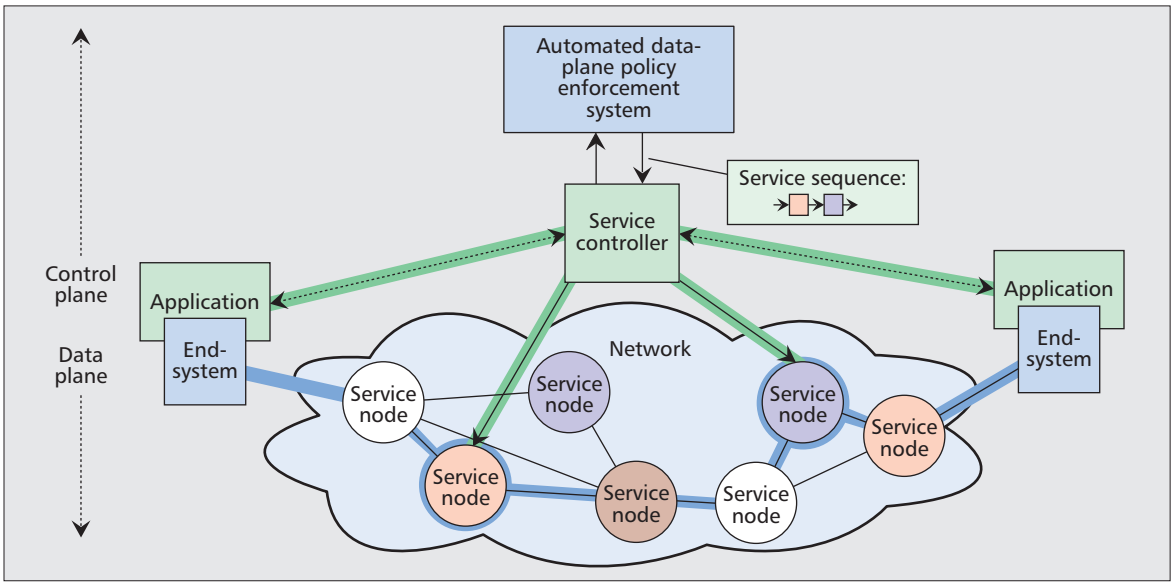

Figure 4. *Network service architecture.*

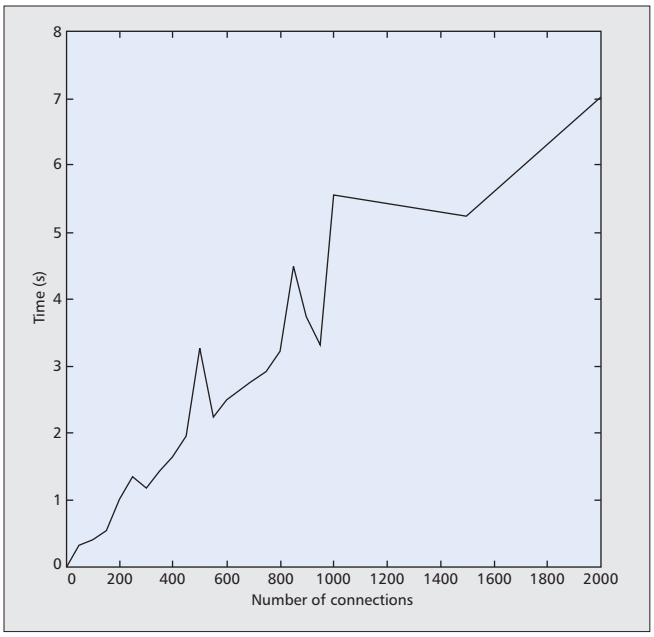

Figure 5. *CPU time (in ms) taken by the planner for generating and repairing plans.*

their preconditions and actions. Table 3 shows the results of the automated composition implementation of example scenarios discussed in the previous subsection. The decision is purely based on the semantics of the data and communication and the service action. Observe that the automatically composed sequence of services achieves the desired output — encryption of a H.264 video file — even though only the input and output data semantics are provided by the end-system.

Automated Data-Path Policy Enforcement

For evaluating the effectiveness of the automated policy enforcement framework, simulations were run on a network of 96 nodes organized into 12 Autonomous Systems (AS) 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 enforces a maximum of two policies every connection and each policy was generated in a random manner. The polices mostly required the addition of a service to the composed sequence to alter the plan.

Figure 5 shows the time taken by the system to generate and repair the plans. All connections had a maximum of two policies enforced on them. As the number of connections increases, the planner takes more time to repair the plan. For 2000 simultaneous connections, the system takes about seven seconds to repair the plan. The simulation was 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 6 compares the lengths of the service sequence before and after policy enforcement. As mentioned earlier, each AS enforced a maximum of two policies every connection. This resulted in the sequence being repaired if the addition of services according to policy resulted in a wrong sequence. Observe that most of the time the planner tends to add new services (while also rearranging the sequence) to the sequence to meet the connection requirements while enforcing policy at the same time.

Related Work

Data-path packet processing in the network is not a new concept. There are many existing examples that have been implemented in the current Internet, such as NAT [3], VPN [4],

Service	Precondition	Action
Transcoder	Type-Video (HDTV-1080p)	Action (converts to H.264.)
Encryption	Type-Text	Action (encrypts payload.)

Table 2. *Preconditions and actions for the example services.*

Scenario 1. Sequence:Transcoder		
Semantics	*Input*	*Output*
Type	Video-(HDTV-1080p)	Video-(H.264 (176×208)
Delivery	Broadcast	Broadcast
State	Stateless	Stateless
QoS	max_delay, min_delay	max_delay, min_delay
Unit	payload	payload
Scenario 2. Sequence:Encryption		
Type	Text	Encrypted
Scope of Encryption	Payload	Payload
Delivery	Unicast	Unicast
State	Stateless	Stateless
Unit	payload	payload
Scenario 3. Sequence:Transcoding → Encryption		
Type	Video	Encrypted
Type	Video-(HDTV-1080p)	Video-(H.264 (176×208)
Scope of encryption	Payload	Payload
Delivery	Unicast	Unicast
State	Stateless	Stateless
Unit	Payload	Payload

Table 3. *Resulting service sequences for the three scenarios.*

etc. The idea of putting advanced packet processing on the data path of the networks has also been proposed as an inherent feature of the next-generation Internet (e.g., SILO project [5, 6], and the service-centric network architecture [7]).

Composition of protocols and services has been studied in the context of the existing Internet as well as next-generation Internet. Configurable protocol stacks [8] and protocol heaps [9] have been proposed as a solution to statically compose novel protocol combinations. More dynamic approaches have been proposed in [7] and [6], where composition can be performed on a per-flow

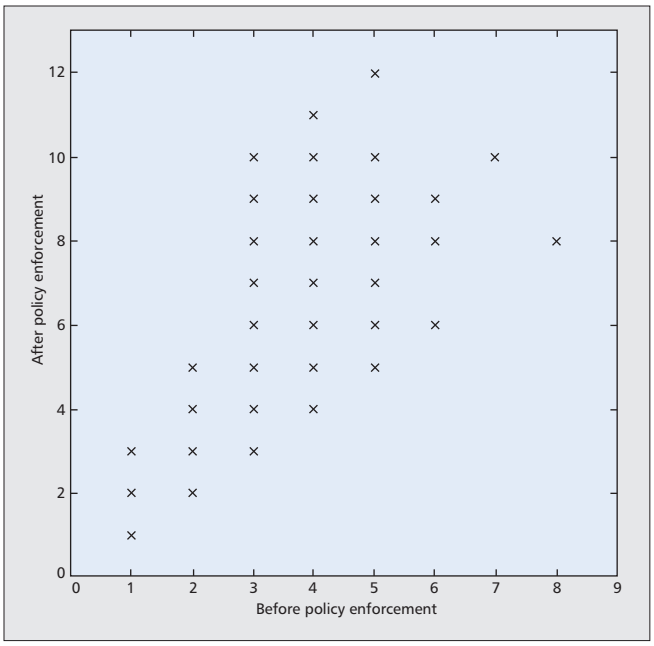

Figure 6. *The number of services before and after policy enforcement.*

basis. The latter uses composition rules and constraints to determine valid compositions. In contrast, work attempts to determine valid compositions by input/output format characteristics of the data and does not depend on exhaustive enumeration of constraints. In research related to automated service composition, various methods have been proposed in the area of web application composition. Most of these methods fall under the category of AI Planning [10] and Theorem Proving [11].

Much research has been done in the area of policy-based network management. Lymberopoulos *et al*. [12] 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. Yoshihara *et al*. [13] 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*. [14] propose an autonomic policy based control service that allows runtime system diagnosis, repair and reconfigurations that helps correct the system in events of conflicts with minimum human intervention.

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. 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.

Summary and Conclusions

The emerging future Internet architecture will need to support a range of different data-path functionalities. To ensure system scalability, these functions need to be composed *automatically* in such a way that policies from different network entities are respected. Our system uses a formal representation of the semantics of data connections, packet processing functions that operate on them, and policies that state constraints and requirements. Using an automated planning tool, connection requests can be re-composed to adjust the connection semantics to meet network policies. The results from our prototype implementation show that this approach can adjust connection requests appropriately and that data-path policies can be enforced. However, this being a centralized design, the system requires a complete view of the network and policies and thus introduces a scalability problem for large networks. A more decentralized approach can solve this problem. This work presents an important step toward a fully autonomic Internet of the future.

Acknowledgements

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

References

[1] T. Wolf, "In-Network Services for Customization in Next-Generation Networks," *IEEE Network*, vol. 24, no. 4, July 2010, pp. 6–12.
[2] A. Gerevini and I. Serina, "LPG: A Planner Based on Local Search for Planning Graphs with Action Costs," *Proc. 6th Int'l. Conf. Artificial Intelligence Planning Systems (AIPS)*, Toulouse, France, Apr. 2002, pp. 13–22.
[3] K. B. Egevang and P. Francis, "The IP Network Address Translator (NAT)," Network Working Group, RFC 1631, May 1994.
[4] B. Gleeson *et al.*, "A Framework for IP based Virtual Private Networks," Network Working Group, RFC 2764, Feb. 2000.
[5] I. Baldine *et al.*, "A Unified Software Architecture to Enable Cross-Layer Design in the Future Internet," *Proc. 16th IEEE Int'l. Conf. Computer Commun. and Networks (ICCCN)*, Honolulu, HI, Aug. 2007.
[6] R. Dutta *et al.*, "The SILO Architecture for Services Integration, Control, and Optimization for the Future Internet," *Proc. IEEE Int'l. Conf. Commun. (ICC)*, Glasgow, Scotland, Jun. 2007, pp. 1899–904.
[7] T. Wolf, "Service-Centric End-to-End Abstractions in Next-Generation Networks," *Proc. 15th IEEE Int'l. Conf. Computer Commun. and Networks (ICCCN)*, Arlington, VA, Oct. 2006, pp. 79–86.
[8] N. T. Bhatti and R. D. Schlichting, "A System for Constructing Configurable High-Level Protocols," *SIGCOMM '95: Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communication*, Cambridge, MA, Aug. 1995, pp. 138–50.
[9] R. Braden, T. Faber, and M. Handley, "From Protocol Stack to Protocol Heap: Role-Based Architecture," *SIGCOMM Comp. Commun. Rev.*, vol. 33, no. 1, Jan. 2003, pp. 17–22.
[10] S. McIlraith and T. C. Son, "Adapting Golog for Composition of Semantic Web Services," *Proc. 8th Int'l. Conf. Knowledge Representation and Reasoning (KR2002)*, Toulouse, France, Apr. 2002, pp. 482–93.
[11] J. Rao, P. Küngas, and M. Matskin, "Application of Linear Logic to Web Service Composition," *Proc. 1st Int'l. Conf. Web Services*, Las Vegas, NV, June 2003, pp. 3–9.
[12] L. Lymberopoulos, E. Lupu, and M. Sloman, "An Adaptive Policy Based Management Framework for Differentiated Services Networks," *Proc. 3rd Int'l. Wksp. Policies for Distributed Systems and Networks (POLICY)*, Monterey, CA, June 2002, pp. 147–58.
[13] K. Yoshihara, M. Isomura, and H. Horiuchi, "Distributed Policy-Based Management Enabling Policy Adaptation on Monitoring Using Active Network Technology," *Proc. 12th IFIP/IEEE Int'l. Wksp. Distributed Systems: Operations and Management*, Nancy, France, Oct. 2001.
[14] N. Badr, A. Taleb-Bendiab, and D. Reilly, "Policy-Based Autonomic Control Service," *Proc. 5th IEEE Int'l. Wksp. Policies for Distributed Systems and Networks (POLICY)*, Yorktown Heights, NY, Jun. 2004, pp. 99–102.

Biographies

TILMAN WOLF [SM] (wolf@ecs.umass.edu) is an associate professor in the Department of Electrical and Computer Engineering at the University of Massachusetts Amherst. He received a Diplom in informatics from the University of Stuttgart, Germany, in 1998. He also received a M.S. in computer science in 1998, a M.S. in computer engineering in 2000, and a D.Sc. in computer science in 2002, all from Washington University in St. Louis. He is engaged in research and teaching in the areas of computer networks, computer architecture, and embedded systems. His research interests include network processors, their application in next-generation Internet architectures, and embedded system security. His research has attracted substantial funding from both industry and the federal government, including an NSF CAREER award. He is a senior member of the ACM. He has been active as program committee member and organizing committee member of several professional conferences, including IEEE INFOCOM and ACM SIGCOMM. He has served as TPC co-chair and general co-chair for ICCCN. He has been serving as treasurer for the ACM SIGCOMM society since 2005. At the University of Massachusetts, he received the College of Engineering Outstanding Junior Faculty Award and the College of Engineering Outstanding Teacher Award.

SHASHANK SHANBHAG (sshanbha@ecs.umass.edu) is a doctoral candidate at the University of Massachusetts, Amherst. He received his B.S. degree from Visvesvaraya Technological University, India and his M.S. degree from the University of Massachusetts, Amherst. His research interests span the areas of network measurement, next-generation Internet architectures, cloud computing and virtualization.