In-Network Services for Customization in Next-Generation Networks

Tilman Wolf, University of Massachusetts Amherst

Abstract

The design of the current Internet lacks in adaptability to accommodate novel network uses and functional requirements. It is therefore important to explore how new services can be introduced into the network infrastructure. We present a novel network architecture that can accommodate the deployment and custom instantiation of such network services. We discuss the motivation for our design and several of the research challenges that arise in this context.

The current Internet has been incredibly successful in providing data communication connectivity between a large number of end systems. This success is evident in how much our society relies on the Internet to provide the means for business, government, and personal communication. One of the main technical reasons the Internet has become the network of choice lies in its network architecture. The use of protocol layers to isolate complexity and ensure interoperability is at the heart of the Internet's design. The *thin waist* of a single globally deployed network layer has ensured that diversity in other layers still results in interoperability.

However, this single fixed network layer has also become the Internet's most challenging limitation. The diversity of end systems connected to the network continues to increase (mobile devices, sensor nodes, etc.), and so do the types of communication paradigms (e.g., content distribution, content-based networking). In addition, the need for security goes beyond what was envisioned at the time of the Internet's design. These trends require an increasing level of adaptability to novel network functions. However, the fixed functionality of the existing network layer does not accommodate such adaptation and thus has limited the deployment of innovative solutions.

Many current problems in networking cannot be solved solely by changing functionality in end systems and leaving the core of the network untouched. Specifically, security, performance, and reliability require support within the network. Therefore, it is imperative that next-generation network designs consider how to integrate adaptability into the network architecture. Such mechanisms for customization are also essential from an economic perspective. Innovation is driven by economic incentives. If network service providers can differentiate their offerings from the competition by adding useful features to their portion of the network, users have the choice to reward such innovation.

In this article we present a network architecture that uses processing services inside the network to provide such custom networking functionality. These *network services* complement novel applications and services that are developed on end systems at the edge of the network. The network service architecture we present addresses some of the key challenges related to customizing network functionality:

- How can custom networking functionality be provided while limiting the overall system complexity?
- How can the data plane and control plane interact to achieve a scalable design for custom network services?
- How can routing be accomplished when communication and computation costs need to be considered?

The remainder of the article is organized as follows. The next section discusses design choices for network service, some background, and the motivation for our architecture. The details of our network service architecture are described in the following section. An overview of research and deployment challenges is then presented. The final section summarizes and concludes this article.

Customization Inside the Network

Innovation in networking is driven by the ability to customize the functionality that is provided by the network. Clearly, there are numerous ways of changing network functionality. The key challenge is to determine a suitable architectural framework such that customization can become an integral part of the network infrastructure.

Customization in the Current Internet

In the current Internet, there are very limited options for customization. Internet Protocol version 4 (IPv4), which is the common protocol among all Internet devices, dictates the structure of addresses, the way packets are processed and forwarded, and so on. Therefore, few aspects of the network can be changed without causing conflicts with the established functionality of IPv4. Thus, the only practical options for customization are:

Customization at higher or lower layers in the protocol stack: The functionality of the network layer (layer 3) is fixed, but there are choices at the transport layer (layer 4). Specifically, one can choose between two transport layer protocols that provide different types of services:

- User Datagram Protocol (UDP) provides unreliable bare-bones connectivity.
- Transmission Control Protocol (TCP) provides reliable connectivity with congestion and flow control.

(There are a few more transport layer protocols, but they are used less frequently and typically target a specific application domain, e.g., streaming media.) Similarly, link layer protocols can be adapted to specific domains (e.g., wireless or optical links).

Customization with middleboxes: The Internet has been augmented by several types of devices that have extended its functionality beyond IPv4 forwarding. Examples include firewalls (which drop unauthorized connection attempts), network address translation (NAT) boxes (which multiplex a private address space into a single IP address), and intrusion detection systems (which inspect payloads to identify and stop hacking attempts). The functionality of these middleboxes needs to be carefully crafted to operate transparently with other IPv4 devices. These mechanisms for customization provide only limited choices. The selection between different transport layer protocols is coarse, and the deployment of middleboxes can only support functionality that can be integrated with IPv4. In addition, deployment of specialized hardware devices for each new customization function is very expensive and does not scale.

Customization as Architectural Principle

To address the shortcomings of the current Internet architecture, it is necessary to introduce customization as a fundamental principle in next-generation network architectures. There needs to be an inherent mechanism to introduce new networking functions into the networking infrastructure. Since it is not possible (or desirable) to define all possible new networking features a priori, customization relies on three specific features:

Programmability: Customization requires some level of programmability in the network such that new networking operations can be specified. The programming environment can be completely general-purpose (e.g., any sequence of instructions is permitted) or restricted to meet certain requirements (e.g., no loops to ensure program termination).

Dynamic deployment: Once new functionality is defined via a custom packet processing program, it needs to be deployed in the network. Dynamic deployment requires that new network services can be instantiated without replacing major hardware or software components in the networking infrastructure.

Selective instantiation: A key requirement for introducing new functions in the network is that they can be selectively employed on traffic. Clearly, not all traffic needs all types of new functions. Thus, the network needs to support a mechanism for choice on which functions are applied to which types of traffic.

The combination of programmability and dynamic deployment provides a basis for introducing new functions, and selective instantiation ensures that these functions are applied to the correct traffic.

It is important to note that some of these features are used implicitly in networks already. For example, many modern router systems use programmable network processors to implement packet forwarding functions. These systems are general-purpose programmable. However, the programmability is not exposed as a feature of the network architecture. Instead, the manufacturer of the router uses programmability to update the implementation of IPv4 (or introduce middlebox functionality). For customization in the network architecture, this level of programmability is insufficient.

Design Alternatives

Customization can manifest itself in a network architecture in numerous ways. We briefly discuss several key design alternatives that need to be considered. This discussion provides the motivation for the design of the network service architecture we present in the next section.

End System vs. Overlay vs. In-Network Services — Novel services that customize the functionality of the network can be placed in several locations in the network. This placement not only affects which types of nodes (e.g., end system workstation vs. in-network router) need to perform service processing, but also impacts how these custom services are represented in the network architecture.

Figure 1 shows three service placements that differ fundamentally in how they are implemented. The scenarios are illustrated using a video distribution example. It is assumed that a high-definition video source needs to stream video to three different clients. The high-definition display on the right can receive unmodified video. The two handheld clients at the bottom require a lower resolution of the video since the wireless links connecting them to the network cannot support the bandwidth necessary for high-definition video, and their processing capacity (and battery power) is too limited to transcode high-definition video to a lower resolution. Therefore, it is necessary to accommodate two video transcoding services that modify high-definition video into a lower-resolution encoding.

There are three fundamentally different ways of placing these transcoding services:

Services on the end system: The simplest scenario of handling services is to colllocate them with the distributed application on the sending or receiving end system. In the case of the video distribution example, the sender performs transcoding for all necessary types of video and transmits each stream. The benefit of this approach is that the network does not need to support any new functionality. The drawback is that services are limited to the end systems involved in the communication.

Services in overlay: Overlay networks use tunnels between end systems to create a virtual topology. Since end systems that implement a service can be placed logically inside the virtual network topology, it is possible to introduce new networking functions without changing anything in the network infrastructure. In the video distribution scenario, an overlay node in the lower left of the figure performs the service processing that modifies the video streams. The benefit of this approach is that services can be placed throughout the virtual network topology of the overlay. The drawback is that service processing is still limited to physical end systems, which requires network traffic to detour (across possibly slow access links).

Services inside the network: If routers can be extended to support service processing, it is possible to place services on nodes inside the network. In the case of video distribution, the transcoding service can be placed on the node where the (multicast) video distribution branches between the high-definition stream and a lower-resolution stream. The benefit of this approach is that it minimizes network resource usage since it does not require transmissions of multiple streams or detours. The drawback is that service placement on routers requires changes in the fundamental network architecture. This example illustrates how the placement of services inside the network can lead to more efficient operation of the network. It is also important to point out that some services can *only* successfully operate in the network. For example, a service related to quality of service (QoS) needs to operate on nodes in the network (e.g., to perform QoS link scheduling) and cannot be implemented successfully on end system nodes. Therefore, the architecture presented in the next section is based on customization services that are located inside the network.

Data Plane vs. Control Plane — The data plane of the network handles the forwarding of actual traffic that is sent. The control plane handles routing protocols, error messages, and similar management functions. In a router system, these two planes are reflected in the data path and the control path. In the data path, packets are received at the input port. The input port processor also implements packet processing functions, including a lookup in the forwarding information base to determine where to send a packet. Then, the packet is forwarded through the switching fabric and buffered on the output port for transmission on the outgoing link. The control path of the router involves the control processor. Routing, control, and error messages are directed to the control processor for processing. Since the data path is typically optimized for high throughput performance, the control path is sometimes referred to as the *slow path*.

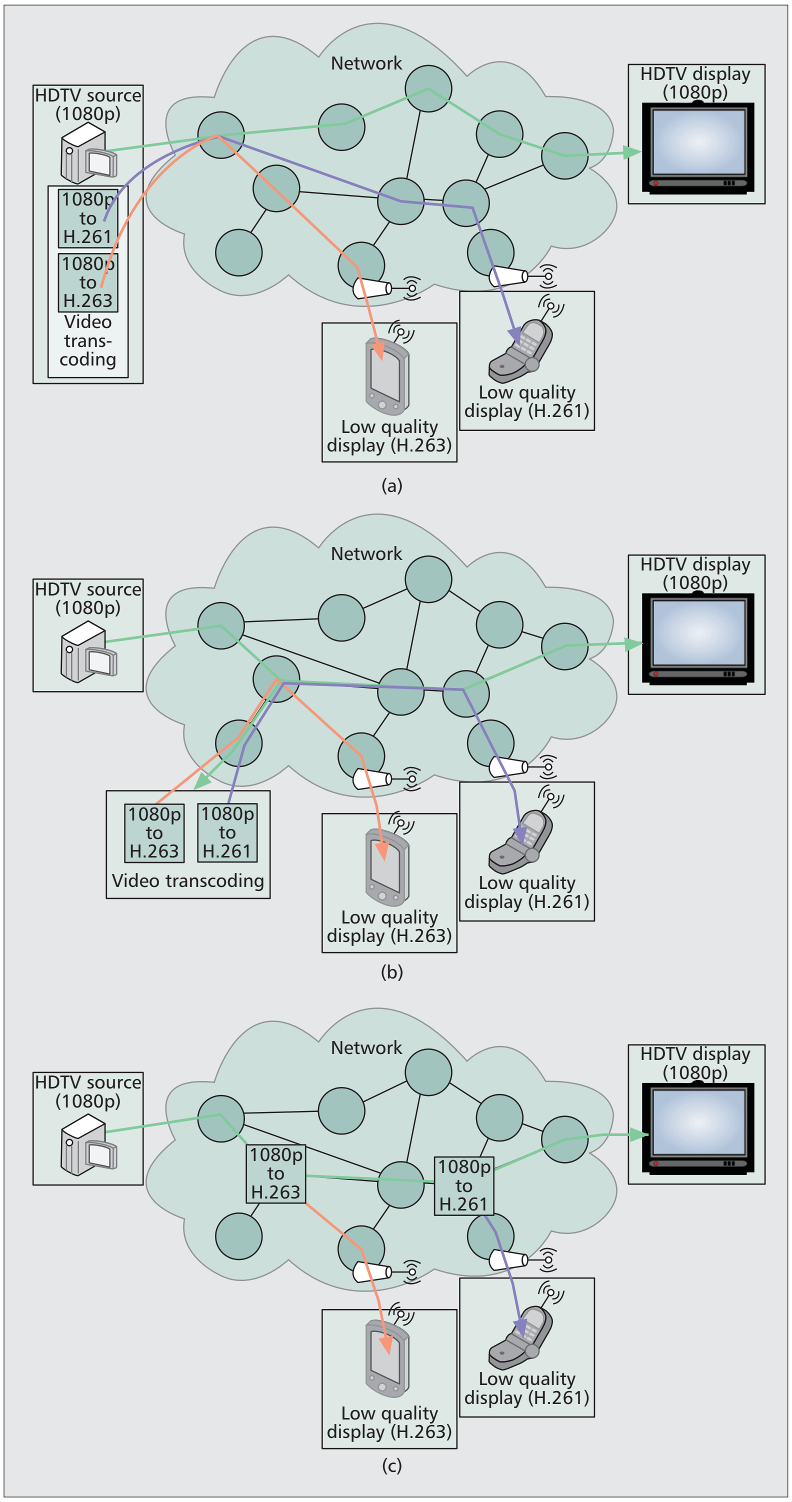

Figure 1. *Alternatives for service placement in networks for a video transcoding example: a) services on an end system; b) services in an overlay; c) services inside a network.*

When considering network customization through in-network services, it is possible to introduce new functionality in the data path as well as in the control path (Fig. 2). An example of a new data path function is a novel intrusion detection process that examines packet payloads. An example of a new control path function is a novel routing protocol that can adapt quickly to link failures.

To ensure that innovative functionality can be deployed in networks, it is crucial that network architecture makes data path customization possible. While some innovation can be achieved with control path customization only, a fixed data path implies considerable constraints (as discussed above for the current Internet). With customization in the data path, the control path needs to implement supporting management functions. These management functions may include selective instantiation of the data path service. The architecture presented in the next section focuses on network customization through novel data path functions. The control path is used for routing and connection setup, and to ensure that the correct set of services is instantiated for each packet.

Generality vs. Manageability — The types of customization that are supported by different network architectures may differ in how general they are. In some scenarios, customization may permit the introduction of any new function. In other scenarios, customization may be based on selection from a given set of functions. The two extremes in this spectrum are:

The current Internet with an ASIC-based IPv4 router: Routers that use application-specific integrated circuits (ASICs) to implement packet forwarding are limited to the functionality that is provided at design time. Such systems cannot be extended to implement any new functions unless the ASIC is physically replaced. A network with such routers is completely static in its functionality. The key benefit of such a network design is its deterministic behavior, low complexity, and easier manageability.

An active network with a general-purpose programmable router: Active networks allow end system applications to introduce arbitrary code for packet processing with each packet [1]. Routers in the network use general-purpose processors to execute this code as the packet traverses the network. Such an approach provides the highest level of generality for introducing new functionality. However, it also causes a high level of complexity and difficulty in managing the network. The introduction of arbitrary code makes it difficult to ensure isolation between connections. It also

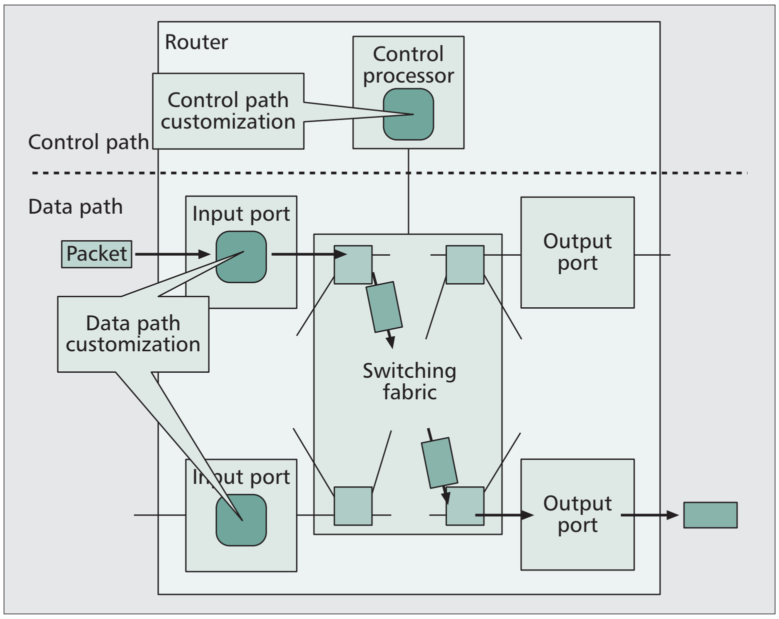

Figure 2. *Customization options in the data path and control path of a router.*

makes it difficult to reason about the behavior of the network since each packet may exhibit different forwarding behavior.

In our architecture presented in the next section, we try to balance the need for new functionality with the need for maintaining manageability. Instead of permitting each connection to introduce new functions, we only permit a custom selection from existing in-network services (where new services are introduced on coarse timescales). As illustrated in Fig. 3, our design of in-network services aims to balance between the generality of an active network design and the manageability of the current Internet.

Network Service Architecture

We present a network architecture that uses network services as basic functional blocks. By composing different combinations of services, network functionality can be customized for different types of traffic. In addition, the introduction of new services permits continuous adaptation to new network uses. We discuss the general ideas behind network services, how they can be implemented in a network architecture, and how they can be used.

Network Services

We use the term *network service* to represent any networking function that is implemented on routers or end systems. In our architecture we decompose the traditional protocol stack into these network services and permit custom composition of services. This process is illustrated in Fig. 4. In the conventional Internet protocol stack, a number of functions are implemented (or have been proposed for implementation) in each layer. Network layer functions are implemented on routers throughout the network and include multicast, QoS scheduling, and others. The transport and application layer functions are typically implemented on end systems and include reliability, flow control, intrusion detection, and more.

Instead of restricting the placement of these functions based on their layer, the network service architecture considers each function as an independent network service. These network services are placed in a pool from which they can be selected arbitrarily. As illustrated in Fig. 4, a connection request can create a custom service composition (in the case of the example, multicast followed by content transcoding). New functions can be introduced by adding new network services to the pool of available choices. As discussed in more detail below, it is not necessary that all routers in the network support all network services (as long as there is at least one system that can perform a particular network service).

The power of this approach of composing network functionality from services is that it balances customization with manageability. We envision that there will be on the order of tens to a few hundred network services and that new network services are introduced at a slow pace (e.g., through Internet Engineering Task Force [IETF] standardization). Thus, new services can be deployed through conventional software update mechanisms for routers. By avoiding end users introducing new services on demand (as was proposed for active networks), it is possible to reduce the management complexity of the architecture. The power of customization in this architecture lies in the very large number of different combinations of network services that can be created.

Network Service Architecture Design

To implement the concept of network services in an actual network architecture, the data and control planes of the network need to be redesigned. Figure 5 shows an outline of the network service architecture for a network consisting of two autonomous systems. In the control plane each autonomous system uses (at least) one service controller. In the data plane service nodes implement forwarding and processing associated with network services. The functionalities of these two components are:

Service controller: The service controller manages the resources within the autonomous system and performs routing of connection setup requests. Routing for connection setup requests consists of finding a path from source to destination as well as determining which nodes perform the set of services requested along the way. Service controllers from neighboring autonomous systems exchange control information for routing and connection setup.

Service node: Service nodes perform forwarding and can

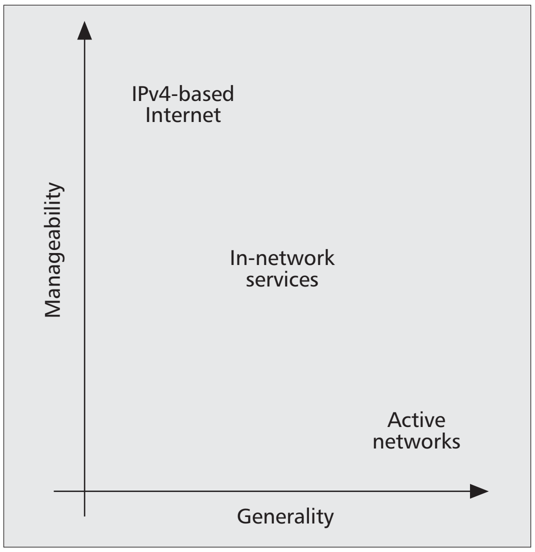

Figure 3. *Trade-off between generality and manageability for different network architectures.*

implement a subset of the available network services. Service nodes inform their local service controller which services they can perform to allow the service controller to make suitable routing decisions. Once a connection is set up, the service controller informs the service node where to route the associated traffic and what (if any) network services to perform.

Using the functionality of service controllers and service nodes, a connection request can be translated into a path through the network that passes service nodes that implement the requested network services (as shown in Fig. 5 for the multicast and transcoding example).

We make several assumptions in this architecture. These include:
- The sequence of services specified by a connection is fixed for the duration of the connection. If a different service sequence is necessary, a new connection needs to be established.
- The underlying infrastructure provides basic addressing, forwarding, and so on (which is being explored in the context of ongoing research on next-generation Internet). Progress in this domain can be incorporated in the network service architecture as it becomes available.
- Connections follow a fixed path during their lifetime.

Fixed routes can be achieved by tunneling or by using a network infrastructure that inherently allows control of per-flow routes (e.g., PoMo [2], OpenFlow [3]). Rerouting in case of link failure can be handled by a renewed connection setup request.

More details can be found in [4] on the network service architecture and in [5] on a prototype implementation.

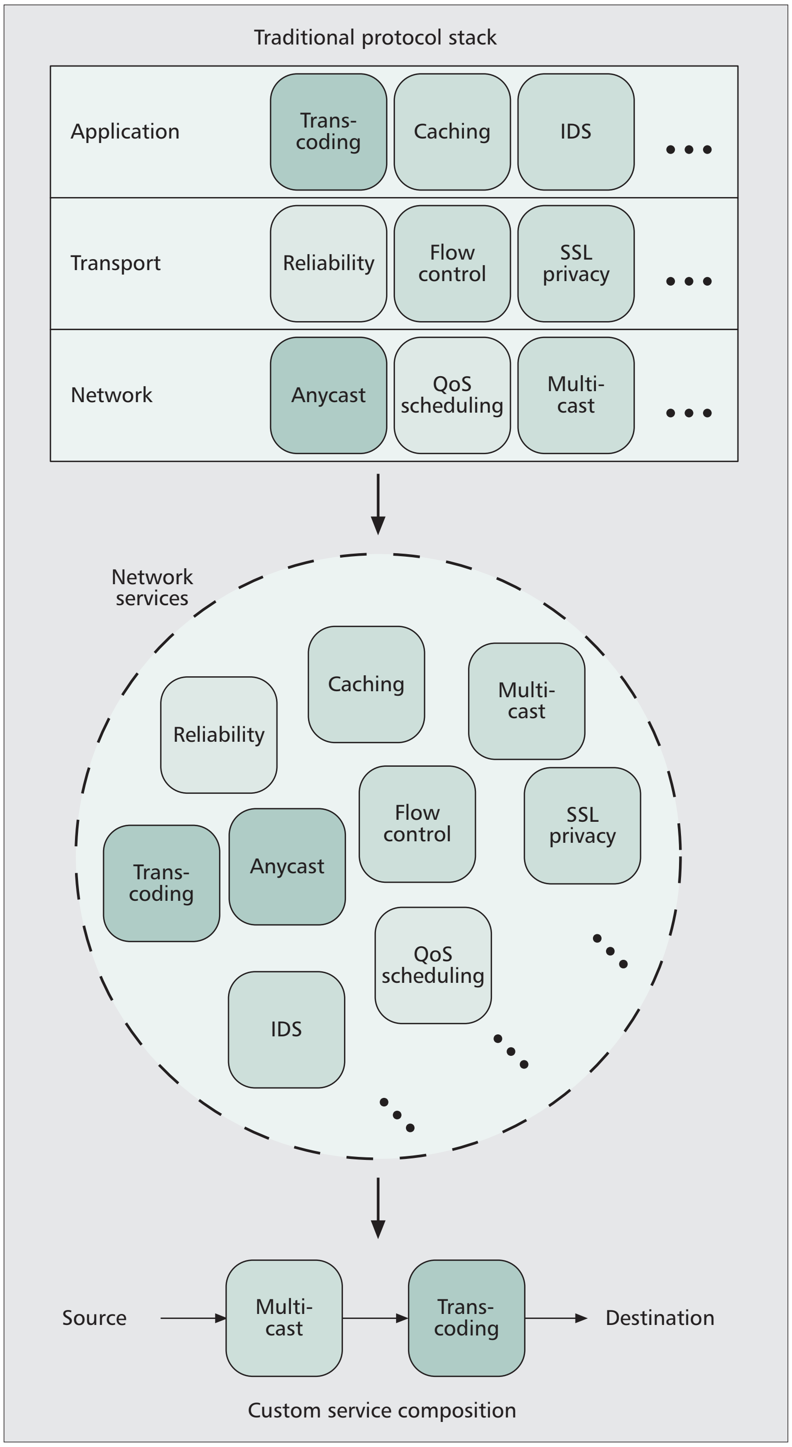

Figure 4. *Decomposition of protocol stack functions into network services and custom composition.*

Interfaces

End-system applications that communicate via the network service architecture use a UNIX socket-style interface. In addition to the destination of the connection request, the sequence of required services needs to be specified. The specification of required services is done using a *service pipeline*. This service pipeline is a concatenation of the sequence of requested network services (and any parameters that need to be passed to them). The service pipeline consists of the following elements:
- Source/destination: Source and destination are specified by their IP address and port number.
- Network service(s): Each network service is specified by its standardized name. Parameters are provided as necessary.
- Concatenation: The source, network service(s), and destination are concatenated to form a linear sequence.

Additionally, a service pipeline can support optional service and branches (e.g., when using multicast). An example of service specifications for video multicast with transcoding is:

```
*:*>>multicast(192.168.1.1:5000,
video_transcode(1080p,H.264)>>
192.168.2.17:5000)
```

The source is not specified (`*:*`), the multicast service branches traffic to two destinations (`192.168.1.1:5000` and `192.168.2.17:5000`), and the latter destination requires a transcoding service (with formats provided as parameters).

The use of service pipelines in the interface to the network service architecture provides a general and extensible method for customization. More details on the implementation of the interface can be found in [6].

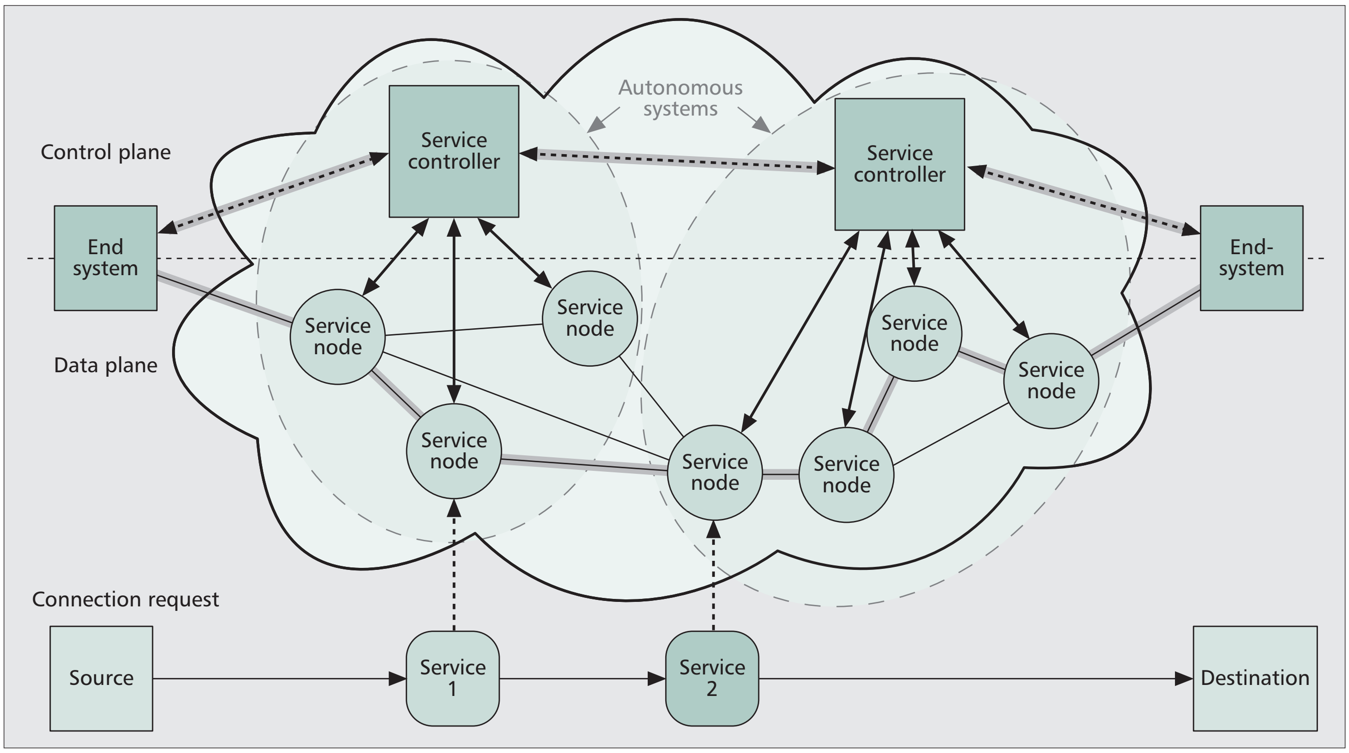

Figure 5. *Network service architecture.*

Network Services in Network Virtualization

Virtualization of the physical network infrastructure is one of the key aspects of next-generation networks. In this context the network service architecture can be integrated in two different ways:

- Network service architecture inside a virtual slice: As virtualization permits multiple network architectures to coexist on the same physical infrastructure, the network service architecture can be deployed as one such slice. Within this slice, connections can specify custom services as described above.
- Network service architecture to specify functionality of virtual slices: The abstractions for specifying and enabling custom functionality in the network can also be used for setting up virtual slices. When a new virtual slice is instantiated, it does not contain any specific functionality. The network service architecture can be used to customize the functionality of this slice. All connections within the slice would use the same service specification provided at configuration time.

In the latter scenario the network service architecture is not explicitly exposed within the slice. Instead, it acts as an abstraction layer to simplify configuration of virtualized networks.

Research and Deployment Challenges

The network service architecture described in the previous section requires solutions to a set of fundamental research and deployment problems. In this section, we briefly highlight several of these challenges and discuss how we have addressed some of them.

Supporting Infrastructure

The network service architecture requires support by the network infrastructure to function. As mentioned above, addressing, forwarding, and similar basic functions are assumed to be available in the network. In addition, it is required that the path of connections can be pinned down in the network (once the route has been determined by network service controllers). This functionality can be provided by tunnels or other network architectures that perform per-flow routing (e.g., PoMo [2], OpenFlow [3]).

On end systems, some level of support for custom network services is necessary. Depending on what functionality is pushed into the network, there may be a need for custom end system protocol stacks. Ongoing research on customization of protocol stacks can complement the network service architecture (e.g., SILO [7]).

Connection Request Routing

One of the key challenges in the network service architecture is to determine how to route connection requests. Conventional approaches of shortest path routing between source and destination are not suitable since the shortest path may not encounter service nodes that provide the requested services. Even if these services are provided along the shortest path, it is not possible to ensure that the path is optimal when factoring in the cost of network service computation. A slightly longer path may traverse a node that can perform the necessary network service processing considerably faster. Therefore, it is necessary to develop novel routing algorithms that can consider both communication and computation costs. We have developed such algorithms for both centralized and distributed implementations. In both cases a single metric is used to represent the cost of communication and processing (e.g., delay). Alternative routing approaches have been proposed in [8].

The centralized algorithm for finding the optimal path between source and destination while traversing nodes that can provide specific services is based on an extended graph representation of the network. The extended graph consists of layers, where each layer is a copy of the entire network. The first layer, which contains the source node, handles traffic before any service is performed. The first layer is connected to the second layer with edges on corresponding nodes where the first requested network service can be performed. The second layer then represents communication after the first service is performed. This layering is continued until the last layer (i.e., after the last network service processing is performed), where the destination node is placed. Then any shortest path algorithm is used (e.g., Dijsktra's algorithm) to find a path from the source in the first layer to the destination in the last layer. Since transitions between layers represent service processing steps, the shortest path in the extended graph determines where to place network services and how to route between them. A detailed discussion of this approach can be found in [9]. The algorithm provides an optimal solution, but requires a complete view of the entire network. Such a global view is unrealistic for an Internet-scale network. Therefore, we use this algorithm only within autonomous systems, where it is reasonable to assume that the service controller is aware of all nodes and links.

A more scalable distributed algorithm for routing with network services is based on an extension of distance vector routing. Instead of simply exchanging the cost of reaching a destination using a distance vector, this algorithm exchanges a *service matrix*. This service matrix contains information on the cost of reaching a destination while performing a certain set of services along the path. The matrix has as many columns as there are service configurations. Since this number increases combinatorially with the number of services, we have developed an approximation that only uses information about the cost of performing a single service along the path. More details on this routing algorithm can be found in [10]. The aggregation of routing alternatives into a single matrix entry using an extension of the Bellman-Ford equation ensures that only a fixed amount of information needs to be exchanged between neighboring nodes. Thus, this distributed algorithm can scale to large deployments. We use this algorithm for routing between autonomous systems (i.e., between service controllers).

Network Service Composition

The network service architecture allows arbitrary composition of network services. Clearly, there are combinations that are not desirable since they do not lead to a useful communication setup (e.g., use of a reliability service on the transmitting end system's side without a matching service on the receiving side). To assist applications in using the network service architecture, we have explored the use of tools to support network service composition. These tools represent the semantics of network traffic and the operations network services perform on these semantics (e.g., transcoding requires an input stream of a certain format and generates an output stream of a different format).

There are two ways of using this composition tool:

- Verification of a service pipeline: Given a service pipeline provided by an application in a connection setup request, the tool can verify that the sequence of requested services provides a suitable communication setup.
- Automated composition: Given the semantics of traffic that is sent by the source and given the required semantics of traffic received by the destination, the tool can automatically compose a suitable sequence of network services.

The latter functionality, which is considerably more complex than simple verification, is based on the use of logic reasoning heuristics. More details can be found in [11].

Summary

The functionality of the Internet needs to adapt to emerging trends in network use and demands for security, performance, and reliability. Therefore, the deployment of custom functionality is an essential aspect of next-generation network architectures. We have presented a network service architecture that allows the introduction of novel network services and a custom instantiation of these services for each connection. We have discussed the motivation for the design of this architecture, and how the data plane and control plane interact. We have also presented an overview of research questions in this domain, including the problem of routing in the presence of service processing requirements.

In addition to these issues, it is important to note that the network service architecture is part of a broader networking ecosystem. There are general questions about the design of routers that can provide high-performance processing, the economic incentives and economic models for such an architecture, the standardization and deployment process of the proposed architecture, and so on. These issues are being addressed by many researchers and practitioners in the broad context of next-generation network research. We believe that the network service architecture is an important contribution in this effort toward a more flexible next-generation Internet.

Acknowledgments

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

References

[1] D. L. Tennenhouse and D. J. Wetherall, "Towards an Active Network Architecture," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 26, no. 2, Apr. 1996, pp. 5–18.

[2] K. L. Calvert, J. Griffioen, and L. Poutievski, "Separating Routing and Forwarding: A Clean-Slate Network Layer Design," *Proc. 4th BROADNETS*, Raleigh, NC, Sept. 2007, pp. 261–70.

[3] N. McKeown *et al.*, "OpenFlow: Enabling Innovation in Campus Networks," *ACM SIGCOMM Comp. Commun. Rev.*, vol. 38, no. 2, Apr. 2008, pp. 69–74.

[4] T. Wolf, "Service-centric End-to-End Abstractions in Next-Generation Networks," *Proc. 15th IEEE ICCCN*, Arlington, VA, Oct. 2006, pp. 79–86.

[5] S. Ganapathy and T. Wolf, "Design of a Network Service Architecture," *Proc. 16th IEEE ICCCN*, Honolulu, HI, Aug. 2007, pp. 754–59.

[6] S. Shanbhag and T. Wolf, "Implementation of End-to-End Abstractions in a Network Service Architecture," *Proc. 4th Conf. Emerging Net. Experiments Tech.*, Madrid, Spain, Dec. 2008.

[7] R. Dutta *et al.*, "The SILO Architecture for Services Integration, Control, and Optimization for the Future Internet," *Proc. IEEE ICC*, Glasgow, Scotland, June 2007, pp. 1899–1904.

[8] L. Xiao and K. Nahrstedt, "Minimum User-Perceived Interference Routing in Service Composition," *Proc. 25th IEEE INFOCOM '06*, Barcelona, Spain, Apr. 2006.

[9] S. Y. Choi, J. S. Turner, and T. Wolf, "Configuring Sessions in Programmable Networks," *Proc. 20th IEEE INFOCOM*, Anchorage, AK, Apr. 2001, pp. 60–66.

[10] X. Huang, S. Ganapathy, and T. Wolf, "A Scalable Distributed Routing Protocol for Networks with Data-Path Services," *Proc. 16th IEEE ICNP*, Orlando, FL, Oct. 2008, pp. 318–27.

[11] S. Shanbhag *et al.*, "Automated Service Composition in Next-Generation Networks," *29th IEEE ICDCS*, Montreal, Canada, June 2009, pp. 245–50.

Biography

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 an M.S. in computer science in 1998, an 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.