Automated Service Composition in Next-Generation Networks

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

***Abstract*—Dynamic composition of protocol features allows applications to establish connections with custom communication characteristics. Automatically computing possible compositions and checking given compositions requires a common framework for expressing application needs, service features, and system characteristics. In this paper, we present such a framework that is based on situation calculus. We show that the automated composition problem can be reduced to an AI Planning problem. We further illustrate the effectiveness of this approach with several examples.**

I. Introduction

The combination of network protocol features used for communication determines the functionality, quality, and performance of a data exchange. Distributed applications have widely differing needs and therefore it is important to consider what choices are available within a network and how to pick a suitable combination. In the current Internet, these choices are very limited and typically come down to a decision between TCP, which provides reliability at the cost of delay, and UDP, which provides low latency with potential packet loss. As the networking community is attempting to design the next-generation Internet [1], several network architectures have been proposed that permit fine-grained control of protocol features.

One such approach is the *network service architecture* we have developed in prior work [2], [3]. In this design, protocol features are expressed as network services that operate in the data path of the network. When setting up a connection from one end-system to another, an application can specify the services that need to be performed as part of the communication (e.g., encryption, reliability, QoS scheduling, etc.). We have developed a socket-like interface to communicate these requirements between the application and the network [4]. Using a distributed routing algorithm, we can determine the optimal placement of services within the network [5]. One main challenge we face in our work (and that has also been encountered by related projects) is the question of how to automate the composition of network service (or protocol features in other contexts).

In this paper, we propose a novel way of formalizing automated service composition as an AI planning problem making use of situation calculus [6]. Situation calculus is a logical language designed for representation and reasoning about action and the changes brought about by the action in the system. In this context, actions are equivalent to the services implemented in the system. For this purpose, we make use of Golog [7] [8], a high-level logic programming language built on top of situation calculus. We also propose a general framework for the representation of application needs, service features and system characteristics. A number of Semantic Web [9] markup languages have been proposed for describing the properties and capabilities of web services as well as the communication level description of the messages and protocols required by these web services. Semantic markup languages enable the design of a generic framework that is scalable and flexible and are devoid of any interoperability issues. For the purpose of representation of service features, system characteristics and data semantics, we make use of the W3C Web Ontology Language (OWL) [10].

In particular, our contributions are:

- A common framework for solving the service composition problem
- An automated solution to check validity of service compositions and to automatically derive service compositions using AI Planning.
- An illustration of the operation of the proposed approach using three specific examples.
- A prototype implementation of dynamic service deployment on a commercial Cisco router.

The remainder of the paper is organized as follows. Section II discusses some related work in this domain. Section III describes the formal framework in which we specify service composition. The automated composition process is discussed in Section IV. Example scenarios are shown in V, and the prototype is discussed in VI. Section VII concludes this paper.

II. Related Work

Composition of protocols and services has been studied in the context of the existing Internet as well as next-generation Internet architectures. Configurable protocol stacks [11] and protocol heaps [12] have been proposed as a solution to statically compose novel protocol combinations. More dynamic approaches has been proposed in [2] and [13], where composition can be performed on a per-flow basis. The latter uses composition rules and constraints to determine valid compositions [14]. In our work, we attempt to determine valid compositions without explicitly enumerating all possible mutual constraints between pairs of protocol features.

In research related to automated composition of web services, various methods have been proposed, a majority of which fall under the category of AI Planning and Theorem Proving. McIlraith et al. [7] [15] [16], propose the use of Golog, a high-level logic programming language based on situation calculus for specification of user requests and constraints. Furthermore, a web service is considered to have its own preconditions and an action that changes the state of the system. Both preconditions and actions are logically represented through situation calculus.

PDDL(Planning Domain Definition Language) is used by McDermott et al. [17] by translating service descriptions in DAML-S [18] to PDDL format. They also introduce a feature called *value of an action*, that enables to distinguish state changes brought about by the execution of the service. Rao et al. [19] [20] treat the problem of automated web service composition as a theorem proving problem using Linear Logic [21]. DAML-S, a semantic web service language is used for the external representation of web service semantics and constraints. Internally, services are represented as axioms and proofs in Linear Logic and a LL-Theorem prover is used for the task of automated web service composition.

III. Service Composition

As described in [2], [3], protocol features and communication requirements are decomposed into basic network services that operate in the data path of the network. Examples of such network services are reliable data transfer (as in TCP), firewalls, intrusion detection, multicasting, privacy, authentication etc. By providing different combinations of the network services along the data-path, the network can satisfy different communication requirements from the end-system applications. However, one of the major challenges of implementing such a network is how to decide the appropriate service composition for a particular communication requirement, which is called the "service composition problem" in our paper.

In this section, we first define the service composition problem and then proceed to present a common framework that can be used to represent communication needs, services features, and system characteristics. This framework is later used to automate the composition process.

A. Service Composition Problem

The service composition problem can be stated as follows: Given a set of network services and a description of some particular communication requirement, check the validity of a given composition or even compute the correct composition that achieves the communication requirement.

Formally, assume $S = \{S_1, S_2, ..S_n\}$ is the set of services available inside the network.

1) Validity Check Problem: Given a communication request R and an ordered sequence of services $(S_{x_1} \gg S_{x_2} \gg, ... \gg S_{x_k})$, the sequence is valid if $S_{x_i} \in S$ for $1 \leq i \leq k$ and $R \equiv (S_{x_1} \gg S_{x_2} \gg ... \gg S_{x_k})$.

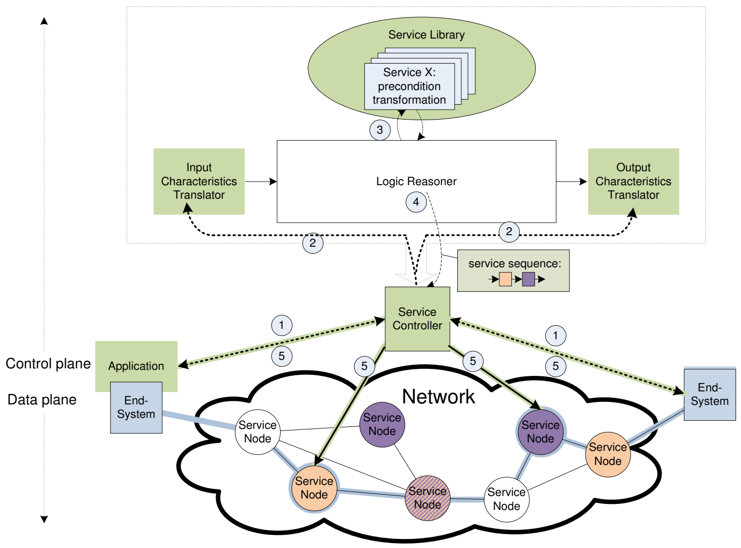


Fig. 1. Service Composition Framework.

2) Automated Composition Problem: Given a communication request R, find an ordered sequence of services $(S_{x_1} \gg S_{x_2} \gg ... \gg S_{x_k})$ such that $S_{x_i} \in S$ for $1 \leq i \leq k$ and $R \Leftrightarrow (S_{x_1} \gg S_{x_2} \gg ... \gg S_{x_k})$.

B. Service Composition Framework

As described in [14], a straightforward solution for the service composition problem would be to first build a comprehensive dependency graph based on a set of rules and precedence constraints between the services. And then use this dependency graph to validate and correct the service composition sequence. However, this solution is not scalable or efficient because it requires a centralized view of all the available network services and an enumeration of all possible mutual constraints between any pair of network services.

We introduce a novel approach to solve the service composition problem in a more robust and distributed manner. Instead of focusing on the dependency relationship between services, we believe a description about the services is more essential. From a systematic description of the input and output characteristics of individual services, the composition dependencies between the services can be fully deduced. Followed by this logic, a general framework is designed to solve the service composition problem. As shown in Figure 1, the framework is composed of three major parts: the input/output characteristics translator, the pre-defined service library, and the logic reasoner. The input and output characteristics translators are responsible for translating the communication requirements into the suitable input and output description that can be understood by the logic reasoner. The service library is a collection of descriptions (i.e., the input and output characteristics of the services) about the available services. The logic reasoner is a piece of software that can help to validate a sequence of service composition or to generate a valid sequence of service composition according to the input and output requirements and the service descriptions. The design and implementation details of the framework is presented in Section IV.

The whole framework can be illustrated by the following scenario: Assume that the server of an online IPTV service is transferring a HDTV-1080p movie to an end-system which can only support and display H.264 (176X208) videos. First, the communication requests are sent to the service controller from the applications running on both the server and the end-system (Step 1). In step 2, the requests are sent to the input/output characteristics translators and translated to the input/output descriptions of the logic reasoner. In our scenario, the input characteristics should be: plain HDTV 1080p video, and the output characteristics should be: encrypted H.264 (176X208) video. After getting the input and output, the logic reasoner consults the service library (step 3) and computes a valid service composition sequence as the results (step 4): transcoding (marked in orange)>>encryption(marked in purple). At last (step 5), the service controller will decide where to put the services and setup the whole path for the connection (the routing protocol and algorithm are described in [5]).

C. Data and Communication Characteristics

One issue that occurs in the design of the above service composition framework is: what characteristics are important enough to be examined when describing the input and output characteristics of either the network or any individual service? After an extensive study of the services, applications, and communication paradigms existed in the current Internet, we designed a tree structure (Figure 2) to represent the data and communication characteristics. The tree representation includes the class hierarchies inherent in data and communication characteristics. For example, the different types of payloads (text, video, audio, images) can be classified as "type" under data characteristics. The leaf-nodes in the tree structure represent parameters that can take on different values depending on the service the structure represents. For example, the Type of Compression node indicates the compression algorithm used whereas max_delay and min_delay nodes under class Delay indicate the maximum and minimum acceptable values of delay. Describing the given tree structure in terms of semantic web markup languages is relatively easy since these languages use a similar structure of classes, subclasses and individuals and different relationships between them. Thus, the data and communication characteristics tree is a general representation that can be extended to include new characteristics and can be described by semantic markup languages.

IV. Automation of Composition

In this section, we describe how we make use of our service composition framework to automate the process of service sequence composition. Furthermore, we also discuss how the framework solves the sequence validity check problem.

A. Validity Check of Composition

Let $S = \{S_1, S_2, ..S_n\}$ be the set of services available inside the network. Let $p_1, p_2, ...p_n$ be the preconditions(input semantics) for services and $o_1, o_2, ...o_n$ be the output semantics produced by execution of the services $S_1, S_2, ...S_n$ respectively. Given an ordered sequence of services $(S_1 \gg S_2 \gg, ... \gg S_k), 1 \leq k \leq n$, with I being the semantics of data input into S_1 and G being the semantics of data produced by the sequence (i.e. the goal to be reached), the sequence can be checked according to our framework to be valid if the following two rules hold:

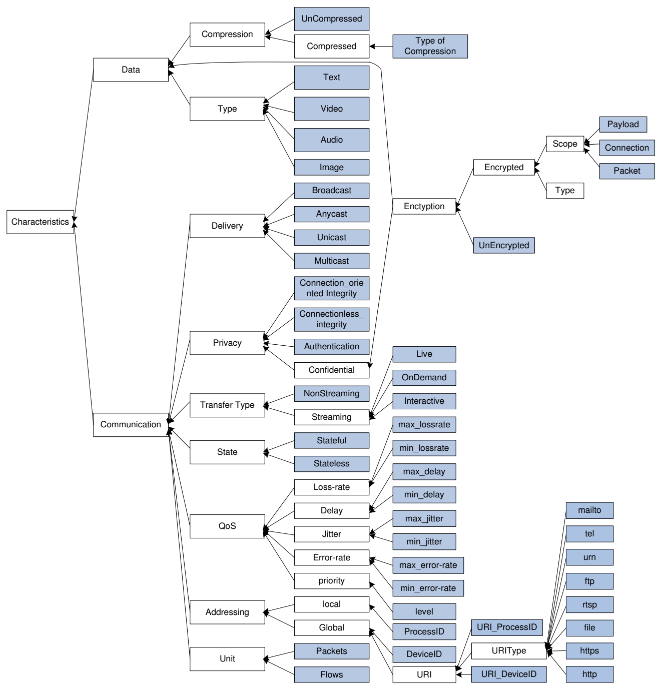


Fig. 2. Semantics for representing characteristics.

1) The preconditions of each service in the ordered sequence are satisfied, i.e. the output produced by a service should satisfy the preconditions of the service that follows in the sequence.
2) The precondition of the first service in sequence S_1 should be satisfied by input semantics I.
3) The output semantics of the last service in sequence S_k is the desired sequence output semantic G.

Each of these rules can be easily checked using our framework. The problem of validity check thus reduces to matching the preconditions of each service with the semantics of the data input to the service.

B. Automated Composition of Services

A straightforward solution to automating service composition problem would be the selection of the set of services dynamically out of the available services and order them in the sequence based on a set of rules and precedence constraints between the services so as to achieve the task. However, this solution involves the enumeration of all possible mutual constraints between pairs of network services.

We take a different approach to the problem of automated service composition, relying more on the connection and data semantics rather than on the precedence constraints between every possible service pair. Data and service semantics enable us to compose complex services to achieve the desired goal automatically. A service can be fully described by the following:

- Preconditions - Preconditions are logical conditions that need to be satisfied in order for the service to be executed. The preconditions include both the protocol and data semantic requirements. For example, an input video stream to a video transcoding service that transcodes from MPEG to H.264 formats has to be MPEG whereas the streaming protocol used can be either RTP, UDP or RTSP.
- Transformation - The function it performs and the transformations it causes in the input data semantics to produce an output with possibly different semantics. The service may or may not change the data stream. For example, a bandwidth monitoring service will not alter the packets in anyway whereas an encryption service will.

Our composition framework allows us to solve the automated service composition problem as follows: Given a set of network services, the description of the preconditions (logical conditions that need to be satisfied in order for the service to be executed), the effect of the service execution on the data semantics, initial and goal states of the data semantics, find a sequential composition of services that achieves the goal while maintaining the precondition requirements of each component service used in the sequence. This problem can be represented as an AI planning problem wherein the services can be regarded as actions, each with its own precondition and effect on global state.

In general, the problem can be represented as follows: Given a set of all actions that an AI planner can perform, A each with its own preconditions and transformation that the action brings about, a set of all possible data semantic states, S, an initial state, $I \subset S$, and a goal state, $G \subset S$, that the planner attempts to achieve, the task is to attempt to achieve the goal state G from the initial state I by performing a series of actions. In the context of network service composition, S is equivalent to the set of all possible semantic states of the data in the network, the initial state I is equivalent to the initial semantics of the data, the goal state G is the outcome of the execution of services that we are trying to achieve and actions, A, are equivalent to network services.

The first step to achieve the proposed automated service composition is to represent the protocol and data semantics as well as the service capabilites (i.e. service characteristics, preconditions etc.) in a formal manner. A number of computer-interpretable Semantic markup languages have been proposed for describing the properties and capabilities of web services as well as the communication level description of the messages and protocols required by these web services. Automated reasoning on semantics described by the markup languages is made possible by mapping the semantic markup (ontology) to a known logical formalism and then using automated reasoners that exist for those formalisms.

As the semantic markup language, we use the W3C Web Ontology Language (OWL) [10] which is expressive enough to enable us to represent service descriptions discussed in the previous section. OWL is a description logic-based language used for knowledge representation by way of ontologies (taxonomic information). OWL ontologies describe data using a set of language constructs like classes, subclasses, individuals and inter-relationships between them along with properties of the individuals. Furthermore, OWL can also express membership restrictions in classes, on data ranges as well as cardinality restrictions. Therefore, OWL can be effectively used to describe network services, as well as data and communication semantics.

For the purpose of AI Planning we use Golog, a high-level logic programming language built on top of situation calculus [6]. Situation calculus is a first-order logical language for reasoning based on action and change. It models dynamical domains as progressing through a series of situations as a result of various actions being performed within the domain. Situation calculus accomplishes goals by combining the situation changing actions. Actions have preconditions which need to be true in order for the actions to be executable. The Golog interpreter is implemented in the Prolog programming language which enables a fairly easy implementation of situation calculus.

V. Examples

To illustrate the effectiveness of our framework, we describe three test scenarios, along with the service requirements and results. In all cases, we have used OWL for semantic markup and Golog for describing services as actions and their preconditions.

A. Scenarios

The three scenarios are:

- Video Transcoding service: Consider an IPTV distribution service that broadcasts HDTV-1080p videos. However, not all subscribers (e.g., PDAs and cellphones) to this service can receive and display high-quality HDTV video. In this case, a transcoding service needs to be executed in the data-path to transcode the video from HDTV-1080p to the format and resolution supported by the subscriber (say H.264 (176X208)).
- Encryption service: An end-system application requests an encrypted communication to upload some documents to a server. This scenario needs a encryption service to be executed in the data-path of the connection to satisfy the security requirement.
- Transcoding + Encryption: Consider an end-system (PDA that supports only H.264 decoding for 176X208 resolution) is requesting a secured Video-On-Demand from a server which can only provide HDTV-1080p sources. In this case, two services, transcoding and encryption, need to be executed in the data-path and the transcoding service need to be done before the encrytpion service.

In all cases, the end-systems just need to specify the data and communication characteristics they intend to send and/or receive through a connection. The services are automatically selected based on this specification and a valid composition is constructed. In the next section, we present the results of the three scenarios discussed.

B. Results

Table I describes the services implemented along with their preconditions and actions. Tables II,III,IV show the results of the automated composition implementation of example scenarios discussed in the previous subsection. We observe from Tables II,III that the requested service is selected by the sequential calculus based reasoning engine. The decision is purely based on the semantics of the data and communication and the service action. Observe from Table IV, 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. Moreover, the correct ordering of services is maintained (since an encrypted video stream cannot be transcoded, transcoding should be done before encryption).

TABLE I
PRECONDITIONS AND ACTIONS FOR THE EXAMPLE SERVICES.

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

TABLE II
RESULT FOR EXAMPLE SCENARIO 1. THE TRANSCODER WAS SELECTED.

Semantics	Input	Output
Type	Video-(HDTV-1080p)	Video-(H.264 (176X208)
Delivery	Broadcast	Broadcast
State	Stateless	Stateless
QoS	max_delay, min_delay	max_delay, min_delay
Unit	packets	packets

TABLE III
RESULT FOR EXAMPLE SCENARIO 2. THE ENCRYPTION SERVICE WAS SELECTED.

Semantics	Input	Output
Type	Text	Encrypted
Scope of Encryption	Payload	Payload
Delivery	Unicast	Unicast
State	Stateless	Stateless
Unit	packets	packets

TABLE IV
RESULT FOR EXAMPLE SCENARIO 3. THE SERVICE COMPOSITION OUTPUT BY THE SYSTEM: TRANSCODING $\gg$ ENCRYPTION

Semantics	Input	Output
Type	Video	Encrypted
Type	Video-(HDTV-1080p)	Video-(H.264 (176X208)
Scope of Encryption	Payload	Payload
Delivery	Unicast	Unicast
State	Stateless	Stateless
Unit	packets	packets

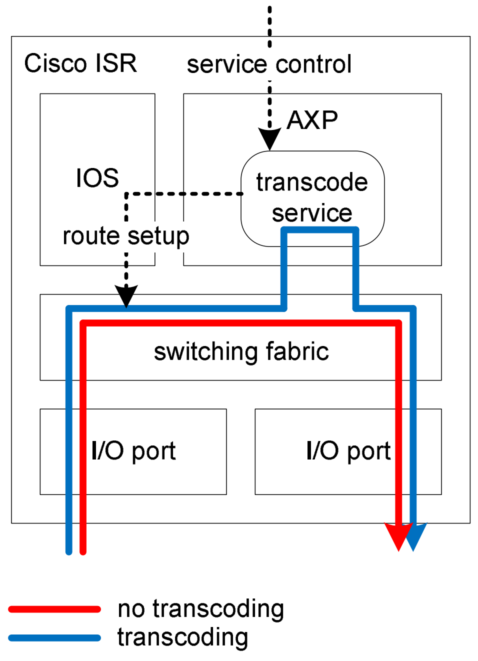


Fig. 3. Prototype Implementation on Cisco ISR with AXP. The AXP implements a video transcoding service that can be turned on and off dynamically.

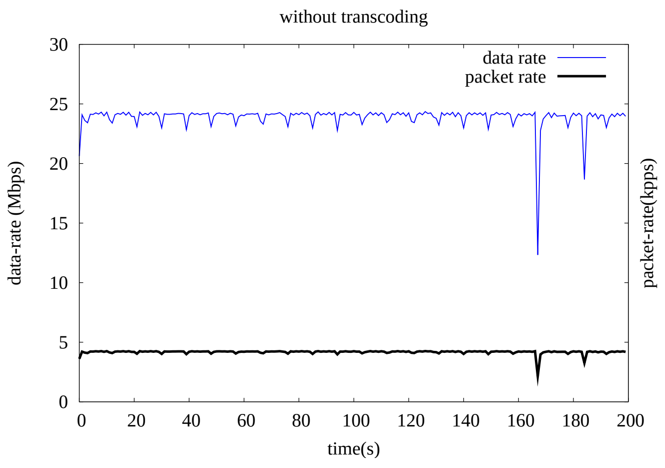
VI. PROTOTYPE

In prior work, we shown the operation of our network service architecture on the Emulab research testbed using a 60-node topology [5]. In this paper, we demonstrate its operation on a commercial router platform. While we only demonstrate a network setup with a single router, this proof-of-concept implementation shows that it is feasible to use off-the-shelf systems for this architecture.

The router used in our prototype is a Cisco ISR 2800 with an Application eXtension Platform (AXP). The AXP is a embedded processor running Linux with specialized I/O and control capabilities to interact with the router system. We have programmed the AXP to implement a video transcoding service that can be instantiated on demand. The service reduces the size of a high definition video stream so it can be played back on a client system that does not have the capability to receive or process a high bandwidth stream (e.g., cell phone).

The prototype system architecture is shown in Figure 3. The AXP hosts the transcoding service. Then a flow needs this service, a control signal is sent by the service controller to the AXP. The AXP then communicates with the ISR's IOS control processor to reroute the flow the AXP. Once the flow streams through the AXP transcoding is performed.

We have implemented the transcoding operation of [Example 1] in the previous section. We have also added monitoring services to record packet and byte counts. The results are shown in Figure 4. Without transcoding, a large amount of traffic is sent. After transcoding the video is reduced in size and thus the amount of traffic reduces accordingly.



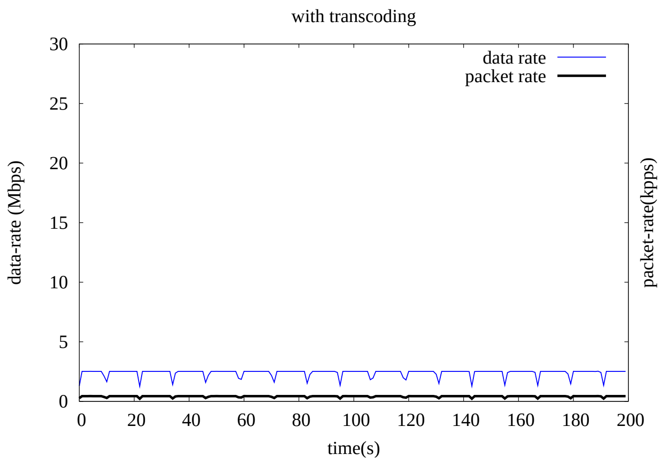


Fig. 4. Transcoding of Video Stream on Cisco ISR. The video resolution changes from 1280x544 to 320x240.

VII. Conclusions

The composition of services is an important problem in any network architecture that permits fine-grained control of protocol features. In this work, we present a novel framework for formally describing network services. Using this framework, we can automate the validation and composition of network service combinations. We show the effectiveness of this approach with three specific examples. We believe that this approach represents an important step towards making highly dynamic and flexible communication configurations an integral part of the next-generation Internet.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. CNS-0626690. We would like to thank Cisco Systems, Inc. for their support related to the ISR system.

References

[1] A. Feldmann, "Internet clean-slate design: what and why?" *SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, Jul. 2007.

[2] 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.

[3] S. Ganapathy and T. Wolf, "Design of a network service architecture," in *Proc. of Sixteenth IEEE International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, HI, Aug. 2007, pp. 754–759.

[4] S. Shanbhag and T. Wolf, "Implementation of end-to-end abstractions in a network service architecture," in *Fourth Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, Madrid, Spain, 2008

[5] X. Huang, S. Ganapathy, and T. Wolf, "A scalable distributed routing protocol for networks with data-path services," in *Proc. of 16th IEEE International Conference on Network Protocols (ICNP)*, Orlando, FL, Oct. 2008.

[6] R. Reiter, 2001.

[7] S. McIlraith and T. Son, "Adapting golog for composition of semantic web services," in *Proceedings of the Eighth International Conference on Knowledge Representation and Reasoning (KR2002)*, Toulouse, France, April 22-25 2002, pp. 482–493. [Online]. Available: mci-son-kr02.pdf

[8] G. De Giacomo, Y. Lesprance, and H. Levesque, "Congolog, a concurrent programming language based on the situation calculus," *Artificial Intelligence*, vol. 121, no. 1–2, pp. 109–169, 2000. [Online]. Available: ConGologAIJ.pdf

[9] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web (berners-lee et. al 2001)," May 2001. [Online]. Available: http://www.sciam.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21

[10] "Web ontology language (owl)." [Online]. Available: http://www.w3.org/2004/OWL/

[11] N. T. Bhatti and R. D. Schlichting, "A system for constructing configurable high-level protocols," in *SIGCOMM '95: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, Cambridge, MA, Aug. 1995, pp. 138–150.

[12] R. Braden, T. Faber, and M. Handley, "From protocol stack to protocol heap: role-based architecture," *SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 17–22, Jan. 2003.

[13] G. N. R. Rudra Dutta, I. Baldine, A. Bragg, and D. Stevenson, "The SILO architecture for services integration, control, and optimization for the future internet," in *Proc. of IEEE International Conference on Communications (ICC)*, Glasgow, Scotland, Jun. 2007, pp. 1899–1904.

[14] M. Vellala, A. Wang, G. N. Rouskas, R. Dutta, I. Baldine, and D. Stevenson, "A composition algorithm for the SILO cross-layer optimization service architecture," in *In Proc. of the Advanced Networks and Telecommunications Systems Conference (ANTS)*, Mumbai, India, Dec. 2007.

[15] S. McIlraith, T. Son, and H. Zeng, "Semantic web services," *IEEE Intelligent Systems. Special Issue on the Semantic Web*, vol. 16, no. 2, pp. 46–53, March/April 2001. [Online]. Available: ieee01.pdf

[16] S. Narayanan and S. McIlraith, "Simulation, verification and automated composition of web services," in *Proceedings of the Eleventh International World Wide Web Conference (WWW-11)*, Honolulu, Hawaii, USA, May 7-11 2002, pp. 77–88. [Online]. Available: nar-mci-www11.pdf

[17] D. Mcdermott, "Estimated-regression planning for interactions with web services." AAAI Press, 2002, pp. 204–211.

[18] A. Ankolekar, M. Burstein, J. R. Hobbs, O. Lassila, D. L. Martin, S. A. Mcilraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, and H. Zeng, "Daml-s: Semantic markup for web services the daml services coalition."

[19] J. Rao and P. Kngas, "Application of linear logic to web service composition," in *In The First International Conference on Web Services, Las Vegas*. CSREA Press, 2003, pp. 3–9.

[20] ——, "Logic-based web services composition: From service description to process model," in *In 2nd Intl. Conference on Web Services*. IEEE, 2004, pp. 446–453.

[21] J.-Y. Girard, "Linear logic," *Theor. Comput. Sci.*, vol. 50, pp. 1–102, 1987.