A New Internet Architecture to Enable Software Defined Optics and Evolving Optical Switching Models

Invited Paper

George N. Rouskas, Rudra Dutta
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
{rouskas, rdutta}@ncsu.edu

Ilia Baldine
Network Research and Infrastructure
Renaissance Computing Institute
Chapel Hill, NC 27517
ibaldin@renci.org

***Abstract*— The design of the SILO network architecture of fine-grain services was based on three fundamental principles. First, SILO generalizes the concept of layering and decouples layers from services, making it possible to introduce easily new functionality and innovations into the architecture. Second, cross-layer interactions are explicitly supported by extending the definition of a service to include control interfaces that can be tuned externally so as to modify the behavior of the service. The third principle is "design for change:" the architecture does not dictate the services to be implemented, but provides mechanisms to introduce new services and compose them to perform specific communication tasks. In this paper, we provide an update on the current status of the architecture and the prototype software implementation. We also introduce the concept of "software defined optics" (SDO) to refer to the emerging intelligent and programmable optical layer. We then explain how the SILO architecture may enable the rapid adoption of SDO functionality as well as evolving optical switching models, in particular, optical burst switching (OBS).**

I. Introduction

For more than thirty years, the Internet architecture has evolved incrementally [13] to address the demands and requirements presented by a continuously changing environment of heterogeneous users, service needs, economic structures, and threats. Typically, a solution for a specific problem is engineered within the constraints of the current Internet architecture. Often, such a solution only applies to a specific context; consider, for example, the recent research on TCP variants for high bandwidth-delay product networks [19], [23], [24], [49], earlier work on TCP over wireless networks [4], [9], [10], [50], and ongoing efforts towards cross-layer optimization [26], [32], [35], [45]–[47]. In other cases, addressing broader needs, such as IP address shortage or security, leads to the development of a more general solution framework, which, however, may violate some of the original principles of the Internet or introduce new elements in its architecture; for instance, network address translation is not consistent with the end-to-end principle, whereas MPLS and transport layer security solutions were introduced at layers 2.5 and 3.5, respectively. With the emergence of optical network technologies, there have been calls for new architectures that take advantage of the new capabilities at both the core [3], [31] of the network and closer to the edges [25] to deliver better performance; these approaches, however, focus primarily on performance issues without explicitly targeting underlying architectural principles.

This work was supported in part by the NSF under grants CNS-0626103 and CNS-0732330.

Recently, as the shortcomings and limitations of today's Internet architecture have become increasingly evident, many in the networking community are convinced that resolving the present "impasse" [2] is impossible without reconsidering the fundamental assumptions and design decisions underlying the current architecture. According to this point of view, it is time for taking a "clean-slate" approach to address fundamental problems and limitations by redesigning the Internet "from scratch," based on new core principles and without being constrained by the existing architecture and protocols [17]. Major clean-slate research initiatives in the US (FIND [18]) and Europe (FIRE [21]) take a two-pronged strategy in clean-slate design: promoting research into new network architectures as well as building an extensive experimental facility for evaluating the new concepts at scale with live traffic. In the US, the NSF-funded FIND program has initiated a research agenda that invites researchers to think carefully about the requirements for the global network in 15 or 20 years, to formulate a vision [41] for the future, and to take steps to realize this vision by devising clean-slate designs unhindered by the current architecture. The FIND program supports a diversified portfolio of research projects with a strong architectural focus; descriptions of these projects are available at the FIND website [20]. NSF has also provided initial funding for the GENI [22] experimental facility designed to act as a catalyst for research on new network applications, services, and architectures. In Europe, FIRE activities are underway within the FP 7 framework, while similar initiatives have emerged in several other countries, including Canada, Japan, and South Korea.

Our 2-year SILO project [37] was among the first batch of projects funded by the FIND program in 2006. The outcome of this project is a new network architecture that overcomes the limitations of layering by introducing the concept of a per-flow "silo" of fine-grained services that can be viewed as a generalization of the traditional layer stack. SILO takes a holistic view of network design with emphasis on facilitating cross-layer interactions, and represents a complete departure from current philosophy and practice; in that sense, it is a truly clean-slate architecture. The SILO architecture and our prototype implementation are described in detail in the next section.

Among recent clean-slate research, there are two projects whose scope extends to include the whole network stack and hence are most closely related to our own project. The first is work on the role-based architecture (RBA) [8], carried out as part of the NewArch project [40]. RBA represents a non-layered approach to the design of network protocols, and organizes communication in functional units referred to as "roles." Roles are not hierarchically organized, and thus may interact in many different ways; as a result, the metadata in the packet header corresponding to different roles form a "heap," not a "stack" as in conventional layering, and may be accessed and modified in any order. The main motivation for RBA was to address the frequent layer violations that occur in the current Internet architecture, the unexpected feature interactions that emerge as a result [8], and to accommodate "middle boxes." The second is the recursive network architecture (RNA) [29], [43] project, also funded by FIND. RNA introduces the concept of a "meta-protocol" which serves as a generic protocol layer. The meta-protocol includes a number of fundamental services, as well as configurable capabilities, and serves as a building block for creating protocol layers. Specifically, each layer of a stack is an instantiation of the same meta-protocol; however, the meta-protocol instance at a particular layer is configured based on the properties of the layers below it. The use of a single tunable meta-protocol module in RNA makes it possible to support dynamic service composition, and facilitates coordination among the layers of the stack; both are design goals of our own SILO architecture, which takes a different approach in realizing these capabilities.

Several years of research in new network architectures (within or outside the FIND program) have produced a broad spectrum of diverse ideas and innovations. More importantly, as the environment (e.g., user requirements, economic factors, threats, etc.) evolves, so will the research community's views regarding the network architecture. In fact, the last observation is at the core of arguments against clean-slate research and in favor of evolutionary approaches that adapt the network to the prevailing environmental conditions [13]. While we agree that the original Internet architecture has been quite successful in accommodating change, we believe that continuing the current practice of applying patches, however innovative, to solve new problems, cannot continue for ever. Instead, we argue that a new architecture, designed with evolvability and adaptability in mind, is a far better proposition moving forward, and that clean-slate research programs offer an excellent opportunity for the community to converge to such a solution. Our goal with the SILO project is not to design the "next" system, or even the "best next" system, but rather a system that can sustain continuing change. To this end, the SILO architecture provides built-in mechanisms to incorporate new services and compose them into silos, making it possible to incorporate new innovations organically and gracefully without the need for patches or add-ons.

The remainder of this paper is organized as follows. In Section II, we provide a brief description of the SILO architecture and of the software prototype we have developed. In Section III, we introduce the concept of "software defined optics" (SDO) and explain how SILO may enable the rapid adoption of an intelligent and programmable optical layer. In Section IV we argue that the SILO architecture may accommodate optical burst switching (OBS) naturally, as well as provide the springboard for the development of a wide range of supporting services and functionality that cannot be easily incorporated in today's Internet architecture. Finally, we conclude the paper in Section V.

II. The SILO Architecture and Prototype

Our design of the SILO architecture was based on three major principles. First, our goal was to maintain the concept of layering, but also to generalize it so as to overcome the current limitations in terms of introducing new functionality. The building blocks of the architecture are *fine-grain services*; each service implements a specific, reusable function of fine granularity (compared to today's protocols that embed complex functionality). Services are composed vertically into *silos* so as to carry out an end-to-end communication task; silos can be thought of as a generalization of the protocol stack concept, but are instantiated on a per-flow basis, allowing each flow (application) to customize its stack according to its own requirements and the properties of the underlying network. Finally, the silo *decouples* layers from services, allowing a service to be introduced at any point in the stack where it is necessary rather than at a specific, predetermined layer.

The second design principle was to facilitate explicitly cross-layer interactions which are awkward and difficult to accomplish in the current architecture. To this end, the definition of a service includes not just the data interfaces to services above or below it in the silo, but also a set of *tuning interfaces*, which we refer to as *knobs*. The knobs are adjustable parameters specific to the functionality of the service (e.g., "compression factor" would be a knob of the "compression" service) with a specified range of values and a well-defined relationship between these values and the perceived performance of the service. These tuning interfaces make cross-service interactions an integral part of the architecture, as it is now possible to employ algorithms that consider jointly the behavior of the services in a silo and *tune* their knobs to the benefit of the communication task at hand.

The third design principle has to do with our desire to "design for change." We accomplish this goal by making sure that we *do not dictate* the services to be implemented. Instead, we provide mechanisms to introduce new services into the framework and *compose* them into silos. Specifically, we

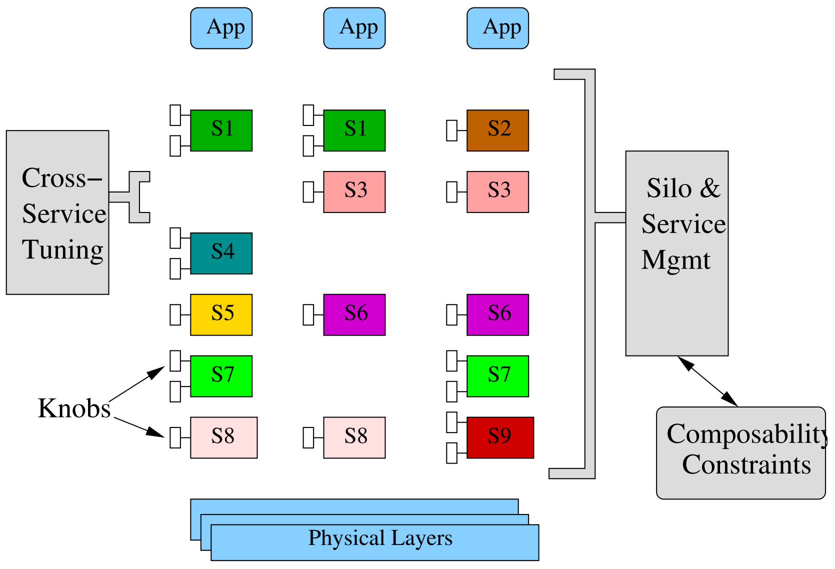

Fig. 1. Generalization of layering

create an *ontology of services* that stores service semantics (e.g., their function and data and tuning interfaces) as well as the relationships among services in the form of constraints (e.g., relative ordering constraints). We have also developed a *composition algorithm* that takes as input service requests (if any) from an application and uses the information and constraints in the ontology to construct a valid silo. Consequently, introducing a new service is straightforward: a developer only needs to provide a description of the service and register it with the ontology for users to be able to include it in their silos.

Figure 1 illustrates the design principles discussed above. The figure shows three applications, each with its own silo of services. It also shows the decoupling of services and layers; for instance, service S6 is at layer 3 of the rightmost silo, but at layer 2 of the middle silo. Each service has its own tuning interface (knobs) that are accessed and tuned by a cross-service algorithm. Finally, the silo and service manager uses the composability constraints to compose services into valid silos. For additional information on the SILO architecture, the reader is referred to [7], [15].

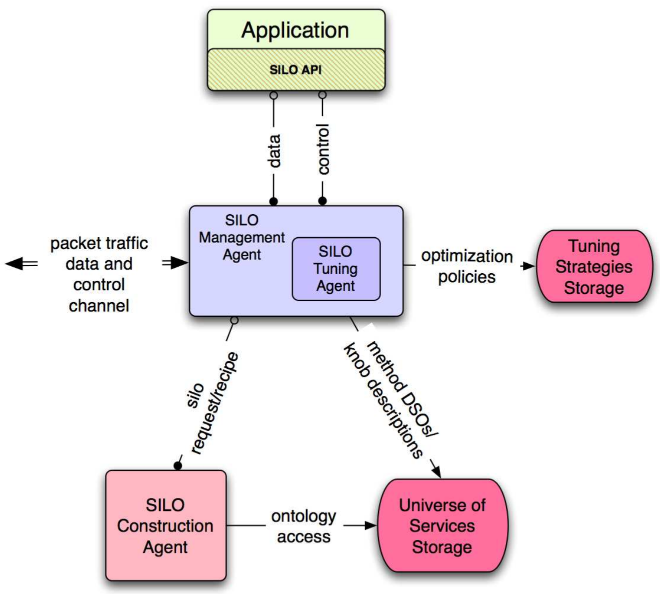

Fig. 2. SILO software prototype architecture

A. SILO Software Prototype

As part of the deliverables of our current FIND project, we have implemented a working prototype which serves as proof-of-concept demonstration of the feasibility of the SILO framework. This prototype, which is publicly available from the project website [37], is implemented in portable C++ and Java as a collection of user-space processes running in a recent version of Linux (although the prototype carries no explicit dependencies on the Linux kernel). Individual services as well as tuning algorithms are implemented as dynamically loadable libraries (DLLs or DSOs). Figure 2 shows the architecture of the software prototype, and its main components are explained briefly in the remainder of this section.

The application communicates with the SILO management agent (SMA) over the SILO API, which consists of header files and library code and serves as a replacement to the common socket interface. Initially, an application creates a *service request*, which describes its communications requirements. The request is passed on to the SILO construction agent (SCA), a major component of the architecture whose responsibility it is to assemble a silo based on an application service request. It utilizes the SILO ontology, an inference engine, and a collection of service composition algorithms we have developed to convert the application request into a *silo recipe* that it returns to the SMA.

Given a silo recipe, the SMA constructs a silo for the application by by dynamically linking in the necessary code for the services (residing in the universe of services storage (USS)), instantiating the state for the new silo, and starting any required execution context.. Once the silo has been constructed, the SMA returns a *silo handle* to the application, which is now ready to transmit/receive data using the silo. The SMA also maintains the silo state throughout the communications session. A component within the SMA, called the SILO tuning agent (STA) is responsible for manipulating the tuning interfaces of the services within a silo in order to optimize either the individual or collective behavior of silos within a single node or among many nodes. The STA selects appropriate tuning strategies from the Tuning Strategies Storage (TSS), after taking into account appropriate user- or administrator-specified policies that are in effect within the USS.

The universe of services storage (USS) serves as the main repository of information about the SILO framework. It contains (1) the ontology that describes relationships between silo services and service interfaces; (2) a database of service implementations which helps the SMA locate the executable code necessary to construct a given silo; and (3) current policy settings which affect the operation of the SILO framework. The USS has a query-based interface, which allows other components of the SILO framework to utilize its functionality. The SILO ontology describes the relationships between SILO services used to create and operate SILOs. It also describes the data interfaces between services as well as service tuning interfaces.

In the terms of implementation, the SILO Management agent is written in C++ as an event-driven loop that maintains multiple silo structures and processes data in transmit or receive direction in individual silos. The SCA is implemented as a stand-alone Java process utilizing the Jena RDF framework API. The USS is implemented as an XML file describing the location of the dynamically loadable code and default parameters for each service. Finally, the ontology was defined using the open source Protégé platform from Stanford and then exported to RDF (Resource Description Framework).

III. Software Defined Optics (SDO)

In today's networks, the physical layer is typically considered as a "black box:" sequences of bits are delivered to it for transmission, without the higher layers being aware of exactly how the transmission is accomplished. This separation of concerns imposed by the layering principle has allowed the development of upper layer protocols that are independent of the physical channel characteristics, but it has now become too restrictive as it prevents other protocols or applications from taking advantage of additional functionalities that are increasingly available at the physical layer. Specifically, in the optical domain, we are witnessing the emergence of what we call *software defined optics (SDO)*, i.e., optical layer devices that are:

1) *intelligent and self-aware*, that is, they can sense or measure their own characteristics and performance, and
2) *programmable*, that is, their behavior can be altered through software control.

Our use of the term SDO is a deliberate attempt to draw a parallel to another recent exciting technology, software defined radios (SDR), devices for which nearly all the radio waveform properties and applications are defined in software [1], [16], [27], [42].

The software logic defining more and more of these SDO devices requires cross-layer interactions, hence the current strictly layered architecture cannot capture the full potential of the optical layer. For instance, the optical substrate increasingly employs various optical monitors and sensors, as well as pools of amplifiers and other impairment compensation devices. The monitoring and sensing devices are capable of measuring loss, polarization mode dispersion (PMD), or other signal impairments; based on this information, it should then be possible to use the appropriate impairment compensation to deliver the required signal quality to the application. But such a solution cannot be accomplished within the current architecture, and has to be engineered *outside of it* separately for each application and impairment type; clearly, this is not an efficient or scalable approach. Reconfigurable optical add-drop multiplexers (ROADMs) and optical splitters with tunable fanout (for optical multicast) are two more examples of currently available SDO devices whose behavior can be programmed according to the wishes of higher layer protocols. Looking several years into the future, one can anticipate the development of other sophisticated devices such as programmable MUX-DEMUX devices (e.g., that allow the waveband size to adjust dynamically), or even hardware structures in which the slot size can be adjustable[1].

In the SILO architecture, all these new and diverse functionalities within (what is currently referred to as) the physical layer will typically be implemented as separate services, each with its own control interfaces (knobs) that would allow higher-level services and applications direct access to, and control of, the behavior of the optical substrate. Hence, the SILO architecture has the ability to facilitate a diverse collection of critically important cross-layer functions, including traffic grooming [14], impairment-aware routing [36], [48], and multi-layer network survivability [28] that have been studied extensively, as well as others that may emerge in the future.

We also note that there is considerable interest within the GENI community to extend the programmability and virtualization functionality that is core to the GENI facility, all the way down to the optical layer so as to enable meaningful and transforming optical networking research. Currently, however, a clear road map on how to achieve such a "GENI-ized" optical layer has not been articulated, mainly due to the lack of interfaces that would provide GENI operations

[1]Note that even as SONET has increased in speed to 10 Gb/s and beyond and its primary function has moved beyond carrying voice traffic, its slot size is still defined by a voice channel; we have shown that the optimal slot size depends on the mix of carried traffic, and should be adjusted as this mix changes over time [33], [34].

access to the functionality of the optical layer devices. We believe that the SILO architecture would be an ideal vehicle for enabling optical-layer-aware networking within GENI, as well as enabling cross-layer research through explicit control interfaces (e.g., such as SILO knobs). Therefore, we are in the process of outlining specific strategies for incorporating the SILO concepts within the GENI architecture whenever appropriate.

A. *Future Research: Deployment in Optical Testbed*

In order to gain wider acceptance of the SILO framework, our goal is to extend the current software prototype and develop a set of services that can provide examples for implementors and at the same time enable experimentation within the framework. Individual services cannot be tested in isolation - they require a "scaffolding" of other services around them that together create a complete and operational ensemble of services constituting working silo. In order to ease the experimentation with the framework we will seek to implement and provide to the researcher community a diverse set of services for these purposes. In the course of this activity, we shall also implement the services required to support the functionality for SDO. Specifically, as a proof-of-concept demonstration of SILO's capability to handle cross-layer functions in a streamlined fashion using well-defined knob interfaces (rather than the current *ad-hoc*, specially engineered solutions), we plan to implement selected SDO functions (including monitoring and sensing services) and related cross-layer functions within an experimental optical testbed facility, as described next.

The Breakable Experimental Network (BEN) is a testbed facility currently under development in the Research Triangle area. This testbed provides experimenters with the ability to control the optical network down to any level they choose, including the fiber transmission itself. BEN will connect RENCI, North Carolina State University, UNC-Chapel Hill and Duke University with dark fiber and will have networking equipment installed at each site dedicated to experimentation, *not* production services. Such a testbed will be invaluable in testing the operation of services for Software Defined Optics.

As an example of deploying the SILO prototype on BEN, consider the classical cross-layer issue of impairment-aware routing in optical networks. We plan to use Dell blade servers with 10G Extended Range (ER) NICs directly attached to dark fiber, integrate them with off-the-shelf equipment that measure optical signal attenuation and chromatic and polarization mode dispersion (CD-PMD), and implement appropriate SILO services to demonstrate that the SILO framework is capable of supporting optical-layer aware routing in an intuitive and flexible manner. We may further utilize the optical fiber switch available at each node of the testbed to demonstrate how optical layer parameters can be dynamically tuned by selectively adding on-demand dispersion compensation and signal amplification for specific connections.

IV. SILO AND OBS

Optical burst switching (OBS) [6], [38], [39], [51], [52] is a switching technology that was introduced a decade ago [31], [44] as a possible solution to carrying large volumes of packet traffic over an optical transport infrastructure. Despite the extensive research effort that has been devoted to OBS, however, except for one prototype implementation [5], [6], the technology has not followed the natural path to implementation, standardization, and wide deployment.

We argue that this unfortunate development is mainly due to two factors. First, OBS does not fit well within the existing Internet architecture. Note that OBS carries IP packets, hence, it operates below the network layer (layer 3); however, OBS semantics are end-to-end within the OBS network, hence it operates above the data link layer (layer 2). Based on this observation, one might be tempted to introduce OBS as layer 2.5 of the Internet architecture, but MPLS [11] is usually considered as occupying layer 2.5. Since it has been suggested that OBS would benefit from label switching [30], it appears that OBS would have to be introduced as layer 2.75 of the architecture, at best an awkward arrangement.

The second challenge has to do with the fact that TCP is not necessarily an appropriate protocol in an OBS network that aggregates TCP segments into bursts for transport over an optical network. Although the performance of TCP over OBS has also been studied extensively in recent years [12], one should keep in mind that the various TCP versions were developed, refined, and optimized specifically for packet-switched networks, hence their applicability to OBS is questionable. We argue that wide deployment of OBS is not possible without a transport protocol customized for this specific transport technology. But the development of such a protocol is inherently a cross-layer design task that cannot be accommodated by the existing network architecture, and has to be specifically engineered using architectural "patches," e.g., as in the case of efforts for adapting TCP to wireless channels. Since applying such patches to the architecture is a challenging and expensive undertaking, it is no surprise that there have been no efforts in this regard within the context of OBS.

Within the SILO framework, on the other hand, OBS may be easily and naturally integrated into the architecture by implementing its component services (e.g., burst assembly, burst scheduling, contention resolution, etc.) and composing them into silos at the edge or core nodes. More importantly, through the design of appropriate control interfaces (knobs) for each service, SILO also makes it possible to develop new transport and routing services that are OBS-aware.

V. CONCLUDING REMARKS

In this paper, we reviewed the SILO network architecture for the future Internet, as well as the software prototype implementation of the architecture. We also introduced the concept of "software defined optics" that pertains to emerging optical layer devices characterized by increasing intelligence and programmability. Finally, we demonstrated how the SILO architecture may enable rapid adoption of functionality related to software defined optics and optical burst switching technology.

REFERENCES

[1] Software defined radio forum, fcusing on open architecture reconfigurable radio technologies. http://www.sdrforum.org.

[2] T. Anderson, L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet impasse through virtualization. *IEEE Computer*, 38(4):34–41, April 2005.

[3] C. Assi, A. Shami, M. Ali, R. Kurtz, and D. Guo. Optical networking and real-time provisioning: An integrated vision for the next-generation internet. *IEEE Network*, 15(4):36–45, July/August 2001.

[4] H. Balakrishnan, , V. N. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. In *Proceedings of the 1996 ACM SIGCOMM Conference*, pages 256–269, August 1996.

[5] I. Baldine, M. Cassada, A. Bragg, G. Karmous-Edwards, and D. Stevenson. Just-in-time optical burst switching implementation in the ATDnet all-optical networking testbed. In *Proceedings of Globecom 2003*, volume 5, pages 2777–2781, San Francisco, USA, December 2003.

[6] I. Baldine, G. N. Rouskas, H. G. Perros, and D. Stevenson. `JumpStart`: A just-in-time signaling architecture for WDM burst-switched networks. *IEEE Communications Magazine*, 40(2):82–89, February 2002.

[7] I. Baldine, M. Vellala, A. Wang, G. N. Rouskas, R. Dutta, and D. Stevenson. A unified software architecture to enable cross-layer design in the future internet. In *Proceedings of IEEE ICCCN*, pages 26–32, August 2007.

[8] R. Braden, T. Faber, and M. Handley. From protocol stack to protocol heap – role-based architecture. *ACM Computer Communication Review*, 33(1):17–22, January 2003.

[9] K. Brown and S. Singh. M-TCP: TCP for mobile cellular networks. *Computer Communications Review*, 27(5):19–43, October 1997.

[10] R. Caceres and L. Iftode. Improving the performance of reliable transport protocols in mobile computing environments. *IEEE Journal in Selected Areas on Communications*, 13(5):850–857, June 1995.

[11] B. Davie and Y. Rekhter. *MPLS Technology and Applications*. Morgan Kaufmann Publishers, San Diego, California, 2000.

[12] A. Detti and M. Listanti. Impact of segments aggregation on TCP Reno flows in optical burst switching networks. In *Proceedings of IEEE INFOCOM*, pages 1803–1812, 2002.

[13] C. Dovrolis. What would Darwin think about clean-slate architectures? *ACM Computer Communication Review*, 38(1):29–34, January 2008.

[14] R. Dutta, A. E. Kamal, and G. N. Rouskas, editors. *Traffic Grooming in Optical Networks: Foundations, Techniques, and Frontiers*. Springer, 2008.

[15] R. Dutta, G. N. Rouskas, I. Baldine, A. Bragg, and D. Stevenson. The SILO architecture for services integration, control, and optimization for the future internet. In *Proceedings of IEEE ICC*, pages 1899–1904, June 2007.

[16] W. Tuttlebee (Ed.). *Software Defined Radio*. John Wiley, New York, 2002.

[17] A. Feldmann. Internet clean-slate design: What and why? *ACM Computer Communication Review*, 37(3):59–64, July 2007.

[18] D. Fisher. US National Science Foundation and the Future Internet Design. *ACM Computer Communication Review*, 37(3):85–87, July 2007.

[19] S. Floyd. HighSpeed TCP for large congestion windows. IETF Internet Draft <draft-floyd-tcp-slowstart-01.txt>, 2003.

[20] National Science Foundation. NSF NeTS FIND intiative. http://www.nets-find.net/.

[21] A. Gavras, A. Karila, S. Fdida, M. May, and M. Potts. Future internet research and experimentation: The FIRE intitiative. *ACM Computer Communication Review*, 37(3):89–92, July 2007.

[22] GENI Planning Group. GENI design principles. *IEEE Computer*, 39(9):102–105, September 2006.

[23] C. Jin, D. X. Wei, and S. H. Low. FAST TCP: Motivation, architecture, algorithms, performance. In *Proceedings of the 2004 IEEE INFOCOM Conference*, pages 2490–2501, Hong Kong, March 2004.

[24] D. Katabi, M. Handley, and C. Rohrs. Tussle in cyberspace: Defining tomorrow's internet. In *Proceedings of the 2002 ACM SIGCOMM Conference*, pages 89–102, Pittsburgh, PA, August 2002.

[25] M. Kuznetsov *et al.* A next-generation optical regional access network. *IEEE Communications*, 38(1):66–72, January 2000.

[26] R. Madan, S. Cui, S. Lall, and A. Goldsmith. Cross-layer design for lifetime maximization in interference-limited wireless sensor networks. In *Proceedings of the 2004 IEEE INFOCOM Conference*, Mar 2005.

[27] J. Mitola. The software radio architecture. *IEEE Communications Magazine*, 33(5):26–38, May 1995.

[28] M. Pickavet, P. Demeester, D. Colle, D. Staessensand B. Puype, L. Depré, and I. Lievens. Recovery in multilayer optical networks. *Journal of Lightwave technology*, 24(1):122–134, Janurary 2006.

[29] The RNA Project. RNA: recursive network architecture. http://www.isi.edu/rna/.

[30] C. Qiao. Labeled optical burst switching for IP-over-WDM integration. *IEEE Communications Magazine*, 38(9):104–114, September 2000.

[31] C. Qiao and M. Yoo. Optical burst switching (OBS)-A new paradigm for an optical Internet. *Journal of High Speed Networks*, 8(1):69–84, January 1999.

[32] V. T. Raisinghani and S. Iyer. Cross-layer feedback architecture for mobile device protocol stacks. *IEEE Communications Magazine*, 44(1):85–92, January 2006.

[33] G. N. Rouskas and N. Baradwaj. A framework for tiered service in MPLS networks. In *Proceedings of IEEE INFOCOM 2007*, pages 1577–1585, May 2007.

[34] G. N. Rouskas and N. Baradwaj. TDM emulation in packet-switched networks. In *Proceedings of IEEE ICC*, pages 1911–1916, June 2007.

[35] V. Srivastava and M. Motani. Cross-layer design: A survey and the road ahead. *IEEE Communications Magazine*, 43(12):112–119, December 2005.

[36] J. Strand, A. L. Chiu, and R. Tkach. Issues for routing in the optical layer. *IEEE Communications*, 39:81–87, February 2001.

[37] The SILO Project Team. The SILO NSF FIND project website. http://www.net-silos.net/, 2007.

[38] J. Teng and G. N. Rouskas. A traffic engineering approach to path selection in obs networks. *OSA Journal on Optical Networking*, 4(11):759–777, November 2005.

[39] J. Teng and G. N. Rouskas. Wavelength selection in obs networks using traffic engineering and priority-based concepts. *IEEE Journal on Selected Areas in Communications*, 23(8):1658–1669, August 2005.

[40] D. D. Clark *et al.* Newarch project: Future-generation internet architecture. http://www.isi.edu/newarch/.

[41] D. D. Clark *et al.* Making the world (of communications) a difference place. *ACM Computer Communication Review*, 35(3):91–96, July 2005.

[42] M. Dillinger *et al. Software Defined Radio: Architectures, Systems and Functions*. John Wiley, New York, 2003.

[43] J. Touch, Y. Wang, and V. Pingali. A recursive network architecture. Technical Report ISI-TR-2006-626, ISI, October 2006.

[44] J. S. Turner. Terabit burst switching. *Journal of High Speed Networks*, 8(1):3–16, January 1999.

[45] J. Wang, L. Li, S. H. Low, and J. C. Doyle. Cross-layer optimization in TCP/IP networks. *IEEE/ACM Transactions on Networking*, 13(3):582–595, June 2005.

[46] R. Winter, J. H. Schiller, N. Nikaein, and C. Bonnet. CrossTalk: Cross-layer decision support based on global knowledge. *IEEE Communications Magazine*, 44(1):93–99, January 2006.

[47] Y. Wu, P. A. Chou, Q. Zhang, K. Jain, W. Zhu, and S-Y. Kung. Network planning in wireless ad hoc networks: A cross-layer approach. *IEEE Journal on Selected Areas in Communications*, 23(1):136–150, January 2005.

[48] Y. Xin and G. N. Rouskas. Multicast routing under optical layer constraints. In *Proceedings of IEEE INFOCOM 2004*, pages 2731–2742, March 2004.

[49] L. Xu, K. Harfoush, and I. Rhee. Binary increase congestion control (BIC) for fast long-distance networks. In *Proceedings of the 2004 IEEE INFOCOM Conference*, pages 2514–2524, Hong Kong, March 2004.

[50] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen. TCP performance issues over wireless links. *IEEE Communications Magazine*, 39(4):52–58, April 2001.

[51] L. Yang and G. N. Rouskas. Adaptive path selection in obs networks. *IEEE/OSA Journal of Lightwave Technology*, 24(8):3002–3011, August 2006.

[52] L. Yang and G. N. Rouskas. Generalized wavelength sharing policies for absolute QoS guarantees in OBS networks. *IEEE Journal on Selected Areas in Communications, Supplement on Optical Communications and Networking*, 25(4):93–104, April 2007.