Report of the NSF Workshop on Software Defined Infrastructures and Software Defined Exchanges

Chairs:
Robert Ricci (University of Utah)
Nick Feamster (Princeton University)

"Beyond the Internet" planning workshop series:
Chip Elliott
Lawrence Landweber
David Farber

February 4–5, 2016

Washington, D.C.

This material is based upon work supported by the National Science Foundation under Grant No. 1546769.

Summary

We are at the dawn of a new era: Software Defined Infrastructure (SDI). Today's relatively static cyber-infrastructures, implemented via hardware with predetermined control systems, are now beginning to morph into fluid, planetary-scale software systems – highly interconnected, deeply programmable, and virtualized within end-to-end slices[1] across many administrative domains. SDI's forerunners include multi-tenant clouds, software defined networking, network functions virtualization, and software defined radios. Individually, each presents major research challenges. But viewed within the broader SDI context, they are simply starting points of a very deep revolution that will reshape our global computing infrastructure.

What is Software Defined Infrastructure? In the not-so-distant past, the process of acquiring and managing computing infrastructure was necessarily very hardware-centric: the norm involved spec'ing, acquiring, and installing physical servers. With the advent of virtualization and cloud computing, we now have a "control plane" for managing this process in software, making more flexible computing infrastructure available to organizations of all sizes, and leading to a paradigm shift in the way we approach computing infrastructures. Similar examples occur in the network domain: software-defined radios (SDR) have made it possible to innovate much more rapidly in a domain that used to be dominated by hardware; software-defined networking (SDN) has opened up the control plane for networks, reducing the dominance of a handful of hardware vendors; network function virtualization (NFV) replaces dedicated hardware appliances with more flexible, innovative software ones. Now that these software-defined infrastructures have transformed individual parts of our computing infrastructure, the next question to ask is how we can take advantage of them in combination. One concrete examples includes various forms of "moving computation to data," such as connecting instruments such as telescopes across the country. Another example is that organizations relying upon virtual networks may wish to install their own (NFV) devices through the network, such as load balancers or intrusion detection systems. Agricultural prediction services might wish to harvest drone sensor feeds from many providers while aggregating the results regionally, etc. SDI opens these possibilities.

In the coming decade, we envision the emergence of a planetary-scale interlinkage of such SDIs, owned and operated by many different organizations, and encompassing trillions of devices, that provides logically isolated, "on demand" global scale infrastructure on an end-to-end basis, with enhanced flexibility and security for new applications. As such multi-domain SDI systems grow and multiply, Software-Defined Exchanges (SDXes) – the "meet-me" points and marketplaces for resources from multiple SDI domains – create the potential to transform the the fundamental structure of the Internet, enabling entirely new classes of research with implications for revolutionary new applications and services.

Background to this report. As the SDI vision has become increasingly apparent, the NSF has sponsored a series of highly successful community workshops to consider the resultant research needs and opportunities. Key NSF workshops to date – including the Workshop on Operationalization of Software-Defined Networks (SDN) in December 2013, the Software Defined Exchange (SDX) Workshop in June 2014, and the Future Research Infrastructure for the Wireless Edge in November 2014 – have made it abundantly clear that experimental research in this area can bring very high payoffs for society and our economy.

The most recent NSF workshop in this series, documented in this report, was held in February 2016 to identify transformative research problems that must be solved before this vision can be realized, with an eye towards research in the 2020–2025 timeframe, and at infrastructure projects that could help to catalyze this research.

[1] A slice is an end-to-end collection of resources, real or virtual, across multiple ownership domains, that are dynamically obtained, used, and released as required by applications. Resources may include compute, storage, networks, sensors, instruments, etc. Software Defined Exchanges help create and manage multi domain slices by supplying compute, storage, and connectivity at multi-domain peering points, and potentially serving as marketplaces that bring together providers and renters of resources.

We envision a world . . . in which all aspects of the planet's cyber-infrastructure form an interconnected, multi-tenant (sliced), and deeply programmable planetary-scale ensemble composed of trillions of devices owned and operated by millions of partially-cooperating, partially-competing organizations. It is conceivable that today's Internet will run in just one "slice" across this infrastructure, with many other novel services populating other slices.

Today's Internet is already beginning to undergo a deep transition from multi-tenant clouds and relatively static infrastructure to a rapidly flexible, deeply programmable new infrastructure based on Software Defined Networks (SDN), Network Function Virtualization (NFV), Software Defined Radios (SDRs), and forward-looking 5G cellular system concepts such as Virtualized Radio Access Networks. We expect this trend to intensify and proliferate in the near future, leading to a rapidly thickening fabric of multi-domain, heterogeneous, edge clouds and interclouds at a scale far beyond today's Internet, and will accelerate as it incorporates mass-market Internet-of-Things devices and large scale cyber-physical systems including cities, autonomous automobiles and ubiquitous drones providing massive "data torrents" from their mobile, high-bandwidth sensors.

As we transition to this emerging vision of planetary-scale SDI, we will need to describe, program, orchestrate, trouble-shoot, and reason about systems that span many different kinds of component devices and subsystems assembled into a wide variety of systems, including many running at a truly planetary scale across millions or billions of programmable devices. Chief among these devices are those that support computation (virtual machines and/or bare metal machines), storage, connectivity, and many types of sensors and actuators. In addition, services will likely be incorporated into these over-arching systems. Furthermore, as is the case today, each of these components (resources) will be owned by someone. The basic issue is of creating and managing reliable, robust end-to-end systems running across a large number of resources residing within many different administrative domains.

What new capabilities will we gain? Can we rapidly assemble custom scientific infrastructure on the fly? Can we transform manufacturing, light up a "sliced" dark factory for a multitude of custom production runs in parallel? Can we program a city, and reprogram responsively if a disaster strikes? Can we channel the overwhelming multiplicity of data torrents while maintaining ownership and privacy? And as always, how can we ensure the security and resilience of these emerging systems?

Fundamental research challenges

Many fundamental research challenges arise with this vision, and a vigorous community debate is now underway regarding the defining features and capabilities of this new SDI-based cyber-infrastructure. Given this very lively debate, it is intriguing that the workshop participants came to express a high level of agreement on three fundamental research challenges:

1) What are the right **abstractions** for representing, programming, troubleshooting, and reasoning about planetary-scale, sliced, multi-domain SDI systems that incorporate an enormous variety of devices and services?

How do we represent and reason about SDI resources? How can we understand, reason about, orchestrate, and ensure privacy and security in such systems? How do we approach fundamental issues of federation and local/global policy enforcement? Today we program individual devices, but surely we need to move to a model of programming these systems as ensembles. How do we bake in security? Are new programming concepts needed? Or new forms of "operating systems" that span many administrative domains? How do we ensure cross-domain decisions have enough data about each domain, without violating the privacy of users or revealing proprietary information? And once we discover innovative and useful new ideas in software based implementations, how can we rapidly instantiate those ideas in new hardware devices for efficiency gains?

2) How can we understand, reason about, troubleshoot, and control the **dynamics** of large-scale SDI systems, and how can we ensure the robustness and resilience of the services they host in the face of unexpected events?

How can we understand control loops at planetary scale, particularly when many layers of adaptive software are interlinked? How do we co-evolve the dynamics of sensor systems, transport, and storage as systems grow and shrink? How can we drill down to troubleshoot malfunctions in multi-domain systems, particularly when some measure of opaqueness is required for each domain? How do we visualize and represent such dynamics to humans? Can we

design algorithms that learn how to robustly control such systems? Can machine learning help? What benefits can be created via market-based approaches to resources and services engaging multiple administrative domains? How do the economics of such systems work?

3) How can we understand, reason about, and manage the **socio-technical aspects** of such systems, including the security, privacy, and data-ownership issues that arise in multi-domain systems that weave together many layers of software?

There is a growing understanding that networks are coming to play an active, rather than passive, role in enforcing or violating the privacy of their users—this points to a new type of relationship between users and networks, but how can users express their privacy needs to their networks in simple ways, and how do they get reconciled with technical and business needs of the network? Many workshop participants expressed their needs to involve real users, real traffic, and real data within their experiments – how can we ensure the appropriate privacy in their research? Can individuals readily understand and control their interactions with such systems? Going further, how can we best explore the fundamental socio-technical issues that arise in these large, multi-domain systems in which many actors interact? How do we visualize such systems? How do we understand systems that span many different legal jurisdictions (e.g. many different national privacy laws)? Can we create mechanisms for "data peering" to provide controlled forms of data movement? Do advances in cryptography provide new techniques for ensuring privacy and security in such environments?

Infrastructure requirements

Although the workshop participants brought many different research perspectives, a few broad themes emerged to drive the infrastructure requirements for supporting such research.

First, researchers will require access to large-scale, software defined infrastructure on which to perform their research; as technology will continue to evolve rapidly over the next decade, the infrastructure must be refreshed often enough to stay comparable with, or somewhat ahead of, mass-market technology. Second, the human aspect: interactions with campus CIOs & US Ignite city infrastructure staff have been beneficial for many researchers, as they have raised many interesting research challenges, and it is desirable that such interactions should continue and expand. Finally, the "multi-domain" aspects of SDI are particularly challenging, and should be explicitly accommodated to support research into federation technologies, multi-domain policy, data peering, etc. From these themes arise two broad classes of infrastructure needed to support SDI research:

1) National-scale **infrastructure suitable for SDI research**, i.e., sliced, multi-domain, and virtualized. It must connect directly to the Internet at as many points as feasible to enable research in combining new SDI technologies into the existing Internet. And it must include an emphasis on underlying technologies that offer radical new capabilities, notably in the areas of large-scale storage, photonics systems, and next-generation wireless access. Alongside this infrastructure, there is a need for software tools that facilitate the full experimentation lifecycle and make it easy for researchers to share data and software artifacts.

2) A set of **5 to 10 Software Defined Exchange (SDX) points** across the United States, which will serve as connection points to the Internet, the "meet me" points for multiple SDI administrative domains, and which provide specific infrastructure support to research with federations, policy, security, data peering, and so forth. Again, this infrastructure must include underlying technologies that offer radical new capabilities in the "data center" space, notably in the areas of large-scale storage and emerging photonics systems.

Starting our voyage of discovery

Today's cyber-infrastructures, implemented via hardware with predetermined control systems, are now beginning to morph into fluid, planetary-scale software systems. This revolution offers the potential for enormous impact on our society, and is creating a wide range of fundamental research challenges. Now is the time to start our voyage of discovery.

Organization of the report

This report was written collaboratively by all participants during the workshop and in the weeks following. Immediately following is a summary of the three keynote talks presented at the workshop; these speakers were

invited by the organizers to give talks that would encourage the participants to think big about future challenges. The majority of time during the workshop was spent in breakout sessions discussing individual topics; reports of the discussions in those sessions (which have been synthesized to produce this summary) were written by the breakout chairs, and can be found next. Appendices at the end contain the list of participants, the workshop schedule, and the call for participation.

Keynote Talks

"SDX: Software-Defined Internet Exchange Points: Where We've Been, and Where We're Going"

Nick Feamster, Princeton University

Nick Feamster gave the opening keynote talk on the Software Defined Internet Exchange Point (SDX) project at Princeton University (http://sdx.cs.princeton.edu/). The talk comprised two parts:

- The past three years of research on building the SDX controller infrastructure, and efforts to make the software controller scalable to size of large commercial IXPs.
- A view ahead of a five-year research agenda on SDXes.

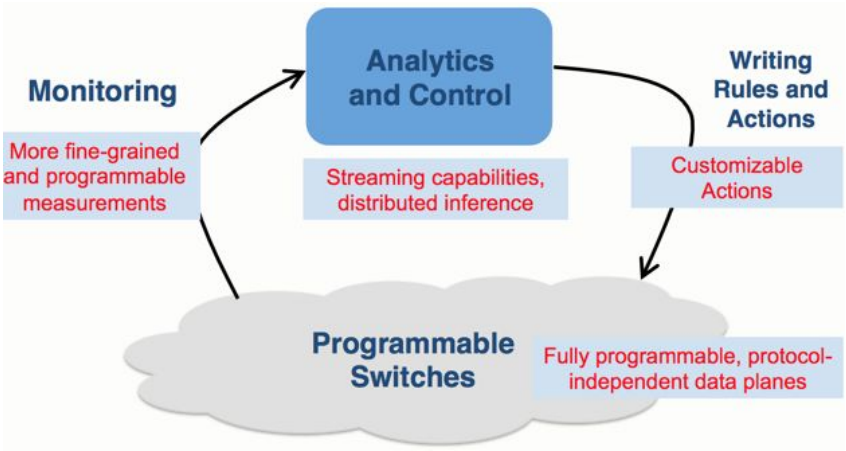
Where We've Been. The retrospective portion of the talk motivated the need for SDXes in interconnected networks where different portions of the network are operated by different (and independent) administrative domains. Although the public Internet is a prominent example of such a network, the talk emphasized that the technologies being developed could also apply to private federated networks comprising independently operated networks.

Some of the initial motivation from the project came from the well-known shortcomings of BGP: its inability to easily control traffic on anything except destination IP prefixes, to control how remote networks forward traffic, or to provide direct, declarative control over how network devices forward traffic. In conventional networks, operators must grapple with BGP configuration arcana (e.g., BGP local preference, AS path prepending) and hope that configuration changes have the right effect. Certain types of control are simply not possible.

The talk presented two examples where SDN-based control at IXPs can provide dramatically more control: fine-grained, coordinated pushback and control of denial-of-service (DoS) attacks and fine-grained control of how traffic enters a network.

The talk also briefly presented evaluation results that show that the current SDX prototype can process BGP updates at the rates seen at the largest commercial IXPs, and can compress forwarding table entries from an entire RIB at a large IXP into forwarding tables in today's commodity hardware switches. The conclusion from this portion of the talk was that (1) SDX applications are compelling and newly possible with emerging infrastructure; (2) technically possible, even with today's (somewhat limited) hardware.

Where We're Going. The second portion of the talk presented a proposal for a five-to-ten-year research agenda for SDX research, keying off of several important technology trends in the SDN "control loop", as shown in the figure below.



The talk highlighted the following trends:

- Fully, programmable, protocol-independent data planes, and mechanisms to program them (e.g., P4)
- More fine-grained, programmable network measurements (e.g., in-band network telemetry)
- Scale-out, distributed streaming capabilities (e.g., Apache Storm)
- Customizable actions in the forwarding plane (e.g., with P4)

In contrast to existing programmable data planes which are relatively fixed-function (e.g., OpenFlow chipsets), emerging technologies, such as those being developed by Barefoot Networks, make it possible to redefine protocols and packet-processing control-flow *at compile time*. P4 allows customization of the parser, the match-action control flow, and the set of actions that the switch can be performed on the packet. At runtime, a controller can populate memory and tables that were defined at compile-time. The talk summarized the current state of affairs for protocol-independent switches: compilation to software switches like Open vSwitch is a reality; within five years, compilation to a hardware switch will be a reality.

The talk closed with several examples of applications that could be deployed at a next-generation SDX—leveraging these emerging technologies that will be realized in the next five years, focusing on three example applications:

- Using in-band network telemetry to attach latency statistics to packets as they travel through SDXes, thus making it possible to pinpoint sources of increased latency, packet loss, or congestion.
- Putting statistical anomaly detection and machine learning “in the loop” for network security and management practices, including (1) identifying malicious ASes based on their BGP-routing behavior and (2) detecting and mitigating DNS reflection attacks.
- Incorporating events from traffic monitors, intrusion detection systems, and other network alert systems into a scalable, distributed stream processing engine (e.g., Apache Storm) to drive real-time routing decisions.

The talk closed by highlighting several operational deployments, including a pilot deployment in the enterprise network at the National Security Agency (NSA) and in a large European internet exchange. The code is available on Github (https://github.com/sdn-ixp/iSDX), and the talk urged researchers and operators to use and build on this software infrastructure for the next generation of research problems, such as those being discussed at this workshop.

"SDN is dead. Long live SDX!"

Marco Canini, Université catholique de Louvain

Marco Canini gave the second keynote talk. The talk started with a retrospective on the status of SDN and the recent research projects on SDXes that laid the starting ground on architectures, capabilities and use cases of SDXes. Despite general interest by network operators, there is evidence that SDXes must mature through much additional research before their disruptive potential can be fully realized. Hence, the second part of the talk focused on what major research challenges exist in the 5-10 years agenda, urging the audience to think about the need for fundamental research that deals with formal foundations about SDXes as well as equally important applied, interdisciplinary research that investigates the role of SDXes in addressing major societal problems.

The talk started by arguing that with SDN the networking research and operators communities have scraped the tip of a large iceberg and it's now time to see what's beneath the water. SDN has reshaped many networks, but in particular most success stories have happened in the datacenter, where it is easier to innovate and deploy.

Motivated by the desire to bring this innovation to wide-area traffic delivery, a natural place for this to occur is at IXPs, which are becoming increasingly prevalent; thus, maximizing the effect onto inter-domain routing of even a single SDN deployment.

Recent projects on Software-Defined Exchanges (SDXes) such as Google's Cardigan, SDX @ Princeton, and ENDEAVOUR have shown feasibility of early ideas in this space. They established ways to incrementally deploy SDN at IXPs, offer more explicit control over inter-domain routing, and address various scalability challenges that exist in large IXPs with hundreds of participants. All of this while maintaining good interoperability with BGP.

The talk argued that the work done so far is necessary "plumbing" and that we need to see more deployments and testbeds in this area. Though future work should stay conscious that technical innovations have to play well within the overall incentives and business aspects of the inter-domain settings, which are inherently more challenging than single-domain settings like datacenters. In support of this argument, the talk briefly presented some anecdotal evidence of the mismatches between opportunities enabled by fine-grained, programmatic traffic control versus current best practices.

The second section of the talk focused on the bold vision set forward by the workshop towards the concept of SDXes that will enable large-scale interconnection of SDIs. The talk discussed a rich research agenda on the road to this vision, highlighting the following:

- Security: Increased Internet security is a goal highly sought after and a desired use case by operators, in particular, detecting and preventing DDoS. Beyond DDoS mitigation, SDXes might enable new architectures that can help by design address network attacks. For example, they provide an opportunity to reconsider the line of research on network capabilities as they might be a way to embed costs into traffic, such that costs can act as a deterrent for attackers.

- Privacy: Beyond improving inter-domain routing, there is a clear expectation of deploying network functions at SDXes such as caches, optimizers, packet scrubbers, etc. This raises questions regarding the privacy of processing traffic at exchanges and the neutrality of SDXes. Who controls the network functions? Who specifies what traffic traverses which network functions? Is there any auditing that we can perform?

- Business Confidentiality: Generally, policies used by ISPs for their peering agreements and route selection are private. SDXes need to provide rich services that can be consumed by participants through APIs without this causing leakage of any confidential information to other participants at the SDX.

- Reliability & Robustness: SDXes will be critical infrastructure that needs to be dependable. Yet this is made challenging by the growing presence of software in networks. Today many experts from the networking and formal methods communities believe that despite their importance, tools for programming and

reasoning about networks are still in a state of infancy.

- Quality-of-Experience (QoE): SDXes are well suited to fill a role in increasing QoE in the Internet. Existing problems and inefficiencies largely stem from the lack of information exchange between application providers and ISPs, which can be detrimental for the user. SDXes might enable a bi-directional flow of information between these parties and act as a trusted optimizing arbiter in the presence of contrasting objectives.

- Marketplace: SDXes prove the unprecedented opportunity to enable an efficient marketplace at the level of transport and routing. There are several forms this can be envisioned: (1) a marketplace for setting up peering on-demand through fast commercial negotiations and connectivity setup, (2) a marketplace for optimizing routing and providing end-to-end guarantees, leveraging the improved network visibility that SDXes have, (3) a marketplace for 3rd party providers of virtualized network functions (e.g., security providers) to tap into the ecosystem.

The talk closed by highlighting the globally disruptive potential for innovation with SDXes: How do we turn SDX into a platform upon which people can build solutions to solve world's problems? Major ongoing trends such as smart grid, smart transportation, smart cities, electronic voting, green economy, online education require better infrastructure as the Internet is ill suited for many new requirements of these technological innovations. The talk posited that SDXes fit into these themes by giving three examples.

1. SDXes for resource fluidity: Connecting hundreds of networks each, an interconnection of SDXes can become a key enabler for better sharing of bandwidth and other resources.

2. Crowd-sourced SDXes: a global network of SDXes can ensure that independent networks can be quickly constructed and operated when times require it (e.g., natural disasters).

3. SDX vaults: in the IoT era with massive data acquisition capabilities, SDXes stand to act as a neutral common ground providing brokering services for privacy-preserving data aggregation and analysis between producers and consumers of such data.

Meeting these goals will require both fundamental research dealing with formal foundations for reasoning about SDXes as well as applied, interdisciplinary work that blends together system prototyping, deployment, testbed operation with use cases and long term vision about the potential impact of SDXes.

A marketplace-driven direction for SDI: Towards an "Open Cloud Exchange"

Orran Krieger, Boston University

Orran Krieger gave the third keynote talk. The talk started by arguing that the current model, where a single entity controls the SDI is a major problem for both industry and research. Focusing on Cloud computing, the talk discussed how the single provider nature of today's clouds result in a opaque cloud that makes research difficult and limits the use cases that the cloud can be used to address. The talk then described how this single provider model stifles competition and innovation both within a cloud and between clouds. Specific examples discussed included:

- Examples of products where a cloud platform provider chose to integrate a service into their platform that impacting the business of a (potentially superior) product by a competitor that was previously built above the platform.
- Price, bandwidth, and functional issues that make switching providers a challenge.
- Lack of visibility/auditing on internal processes and procedures.
- Price challenges for applications that don't fit the intended targets of the cloud.

At a fundamental level, the problem is that the provider's incentives are not aligned with creating an efficient marketplace, but are rather aligned with locking customers into their own offerings. The analogy was made to the pre-internet world, where users were locked into vertically integrated companies like AOL. The question was posed if this is in fact intrinsic to Clouds/SDI, or if an Internet like solution could be developed that would better enable competition, innovation and research.

The talk then argued that a more open model, an Open Cloud Exchange (OCX), is possible, where multiple entities can collaborate and compete in standing up a cloud. The claim was made that, while creating such models can be disruptive and difficult, historically open models tend to be much more successful, with reference to "The Cathedral and the Bazaar" by Eric Steven Raymond, and "The Master Switch: The Rise and Fall of Information Empires" by Tim Wu. This part of the talk concluded with a discussion of how the existence of an OCX could fundamentally change systems research, providing researchers with access to real data, access to real users, access to massive scale. Participating in an OCX, researchers could directly make innovation available to a real community of industry and end-users.

The next part of the talk focused on the viability of create an OCX. Reasons why it is not impractical included: 1) the participation of industry groups locked out of today's vertically integrated clouds, 2) niche markets not addressed by today's clouds, 3) availability of open source, etc… It then described the Massachusetts Open Cloud (MOC), an OCX that is under early development in a collaboration by BU, MIT, Harvard, Northeastern, and UMass with core industry partners including Intel, Lenovo, Brocade, CISCO, and Two Sigma.

The major part of the talk then described the long term research challenges that are either: 1) required to fully realize the OCX model, or 2) enabled by that model. The goal was not to be exhaustive, but rather to identify a number of major examples in each category with some degree of depth. Examples of required research include: 1) marketplace mechanisms (users to describe needs/constraints/preferences, allow providers to describe services, incentives to provide high quality services and/or rich information), the need for Hardware as a Service, the need for Cloudlets, the challenges of software defined storage and networking in an OCX, how to expose rich information in a privacy preserving fashion, how to identify sources of failures. Examples of research enabled, included smart cities, analysis of cloud state for security or optimization, cloud security in an environment with non-colluding entities, new hardware infrastructure, and highly elastic environments.

Breakout Sessions

Overview of the Breakout Sessions

This workshop engaged a large number of highly enthusiastic (read: voluble) researchers and R&E infrastructure operators, nearly all of whom have had substantial prior experience with Software Defined Networks and many of whom have created prototypes of Software Defined Infrastructure. All were boiling over with ideas on SDI, where it is going, and the research challenges that it raises.

The workshop was thus structured primarily as a series of discussion sections, rather than as a series of presentations, in order to give maximum opportunity for participants to speak their minds, argue, and interact. They took full advantage of this opportunity, as the following sections of this report will show.

The workshop chairs assigned participants to specific breakout sessions, based on the contents of their white papers, but of course offered participants the opportunity to switch sessions if desired. Each workshop member thus participated in a series of sessions, and therefore within a kaleidoscopic series of discussions with a ever-changing cast of co-participants. While this leads to some overlap in the session write-ups, it gave rise to a great deal of cross-fertilization across the participants, which was extremely valuable, as we came from a range of backgrounds and perspectives. Here is an overview of the workshop sessions:

This approach proved highly successful in practice, as a number of strongly-shared core beliefs emerged very clearly out of the “kaleidoscope” of sessions. The executive summary has documented these shared beliefs. The following sections provide detailed descriptions of discussion in each breakout session. They were collaboratively written by the session chairs and participants.

Models for Programming and Verifying SDI

Participants: Aditya Akella (chair), Anduo Wang, KC Wang, Rick McGeer, Theo Benson, Dan Kilper, James Chacko, Joe Touch, Mehrdad Moradi, and Srini Seshan.

This section focused on the basic questions of how we will program and verify an SDI system as a whole, rather than as an assemblage of many individually-programmed components. We identified what we believe are four main, intertwined research themes in the general space of programming and verifying SDI. These span low level hardware support/operations all the way up to the high level interfaces exposed to different entities. We also identified two sets of main infrastructure needs to catalyze research along these themes.

Transformative research areas

1. High-level abstractions: Semantics, Evolution, Layering

The vision of SDI and SDX portrays a paradigm where an application can programmatically create and control a collection of diverse types of interconnected resources (compute, storage, networking, data, sensors, cyber physical systems such as vehicles, etc.) within an end-to-end slice that spans multiple administrative domains.

Beyond the apparent need of abstractions to program each individual entity type's specific properties, it is broadly agreed that one or more layers of higher level abstractions are needed to address a number of issues, and these abstractions are expected to evolve with the SDI scope, technologies, and needs over time. For example, an abstraction will be needed to express policies for programming SDI across multiple domains. For another example, an abstraction may be used to express application requirements without dictating specific underlying SDI, thereby allowing lower layer abstractions to more flexibly compose the SDI to meet the requirements. SDX, as an exchange point of multiple SDIs, is expected to be the place where inter-domain resource peering takes place; as a result, an abstraction is in need to express the multitude of potential contractual relationships across SDIs.

Research is needed in systematic ways to express diverse abstractions at different levels and of disparate natures, including data representation methods, and language for transformation across abstractions.

2. Programming model expressiveness/complexity, implications for verification

The point of Software-Defined Infrastructure is an infrastructure that is at once more flexible, controllable, and transparent to user and developer. One important characteristic of this infrastructure is that it is not owned or controlled by the user. At runtime, it is an opaque black box. Thus, it must have *guaranteed* properties of both performance and function. Infrastructure also has limited visibility and debuggability. It's hard to diagnose network problems, and it's hard to diagnose runtime issues on a remote system. Thus, programs which manipulate the infrastructure (e.g., orchestration systems, SDN applications, etc.) should have their infrastructure manipulations verified, to the extent that this is possible – we need to catch bugs statically to the extent that we can, performance and correctness both.

Fortunately, infrastructure configurations ought to be inherently verifiable. The Turing hierarchy has five levels: state-free, finite-state, pushdown, linear-bounded, and Turing-complete, getting stronger as one ascends the hierarchy. But verifiability varies inversely with strength. *Weak models make for strong verification.* Verification of state-free systems is in NP; verification of finite-state systems, at least for safety properties, is similarly in NP (the argument is that a regular expression can be given for a sequence which violates a safety property, and the size of a regular expression is linear in the size of the FSM description). However, verification of push-down automata is P-Space complete and verification of Turing Machines is, of course, undecidable.

It has been shown by a number of authors that OpenFlow rulesets are state-free, and verification is therefore in NP. Similar arguments can be made for various orchestration layers and workflow engines, depending on precise

semantics. *These results imply that the underlying model of computation for configuration of software-defined networking and at least some elements of software-defined infrastructure are state-free or, at worst, finite-state, and therefore that verification of these systems is relatively tractable*. It is, at the least, not undecidable.

The large challenge before the community is then to design configuration models for software-defined infrastructure that preserve the precise and weak semantics of the implementation domain; offer appropriate abstractions of performance characteristics; and nonetheless retain usability and concision.

A negative example and warning comes from the world of the specification of Very Large-Scale Integrated (VLSI) circuits. Programmatic descriptions of VLSI first began to emerge in the 1980's. While some academic high-level description languages, notably the silicon compiler efforts from MIT and UC Berkeley, were semantically valid and well-tuned to the implementation domain, the languages that were adopted in industrial practice were neither semantically faithful to the implementation domain nor amenable to automated verification. These languages (Verilog and VHDL) were designed to make event-driven simulation of the circuits easy. They were *simulator programming languages*, with semantics directly tied to manipulation of the simulator event wheel. They were unmatched to the semantics of the implementation domain, leading to subsequent problems in "circuit synthesis" (aka, compilation) and such dreck as the "synthesizable subset" of Verilog and VHDL (imagine a "compilable subset" of C). Worse, they were Turing complete, and thus descriptions were unverifiable – though the implementation domain was finite-state and therefore verifiable. VLSI designers traded a few weeks of convenience for years of bugs.

Better description languages would have had a core tied to the verifiable implementation domain – a paper-thin abstraction over logic circuits and finite-state machines, the actual objects being designed, with decorators for relevant performance properties such as delays and performance requirements. Simulation directives and programming should have been captured in a simulation harness written in a general-purpose language. Given the era, the choices for the last were not good, and the unfortunate community would have probably been saddled with C or Java; but even those choices were rather better than the truly disastrous one they made.

This history is relevant because, like the early VLSI designers, the SDI community will be dependent on a mix of simulation and formal methods for performance and correctness debugging of our implementations; and because the implementation domain is radically different from a general-purpose computer.

So what are the semantics of the implementation domain? We specifically exclude from this discussion computations at the orchestration and control layers. These, and functional programs such as (say) DNS resolvers implemented on middleboxes, are traditional general-purpose Turing-complete programs implemented on standard computers, and debugged in the usual way. It is the configurations that SDN controllers generate and the configuration instructions which govern the deployment, configuration, and activation of services throughout the infrastructure that concerns us here. The various network services, to the extent that their state is relevant to the verification challenge can be conservatively modeled as finite-state black boxes. The semantics of the domain are therefore:

- Combinational (state-free) transition rules for the switches in the various networks, and/or abstract forwarding rules with delays for the network as a whole
- Non-deterministic FSMs to represent the state and configuration of each component
- An overlying graph structure with delays

This is a first approximation; subsequent investigation will yield a richer semantic domain. We note that the Frenetic language captures some of this, as does the "pipe" high-level language of Click. Of particular interest above these crude semantics are the panoply of issues in programming language design: overlying abstractions beyond the bare FSM level; scalability; and dealing with uncertainty (networks of communicating FSMs have a shared global state; in the real world, assuming the CAP theorem is true, they don't. How do we model this uncertainty?)

3. Instruction Set Architecture for the SDI dataplane

SDN currently supports a small, fixed set of operations to manipulate the fast-forwarding path of a network switch. We can think of these operations as the instruction set of a programmable device and the translation from user/operator requirements to SDN configuration as a compilation process. Naturally this process will need to be greatly generalized as additional elements (computer, storage, etc.) are brought within the programming model. Network Functions Virtualization (NFV) has begun this process, but its programming models are currently very rudimentary, e.g., placement and configuration of firewalls or load balancers.

This begs typical concerns over the instruction set:

- Is there benefit to grouping sequences or sets of basic SDI configurations / functions as a higher-level operation, e.g., to combine the RISC operations into CISC via a kind of microcoding?
- How do we deterministically translate between high-level user/operator intent and these basic instructions on different SDI architectures - i.e., are there opportunities for compiler portability and machine porting?
- Are there operations that we can add to support user/operator intent that would be otherwise difficult to translate? E.g.:
  - Packet replication with copy identifiers, to support multicast/anycast and side-effects such as generating an ARP when a packet arrives (e.g., to support LISP dynamic egress discovery)
  - Packet merging based on min, max, sum, etc. binary content operators
  - Packet “bubble tracks” (inspired by ionization tracks in bubble chambers), i.e., soft state that persists for only a few packet times, that can be used to support packet merging, packet trains, etc.
  - Would there be a benefit to having a portable intermediate set of operations, such that all compilers/verifiers convert user/operator intent to these intermediate codes and a separate translator from these intermediate codes to particular “machine codes” (i.e., like P-code for Pascal). This could increase flexibility for implementations while ensuring stable development tools across the chain.

Note that the use of a RISC-like set of instructions or CISC extensions does not imply that all functions are implemented in the data plane. Some devices may implement more costly instructions or instruction grouping (ala microcode) using a control plane or co-processor element, rather than natively in the data plane.

4. Delegation: slightly smarter physical elements vs. hierarchical control

In SDI, there are some tasks that are repetitive in nature. Existing OpenFlow networks offload all these repetitive tasks to the SDN controller. For example, a basic host discovery protocol requires several rounds of ARP message crafting and exchange between the SDN controller and switches. Similarly, the SDN controller must periodically push broadcast link discovery LLDP messages into the SDN switches to learn new links and their corresponding failures. In these cases, we could design more scalable SDI networks if we push some of these tasks to the virtual or physical resources themselves. In general:

- Designing a flexible boundary between control plane application and data plane processing helps optimizing the network functions and improving the traffic rate.
- These boundaries can be heterogeneous, which means they are potentially different for each SDI.
- Hierarchical control plane is another instance where distributing control plane tasks among physical data plane and distributed controllers are interesting.
- A declarative object-driven language might be useful for achieving the above goals.

Infrastructure requirements

1. A mix of highly programmable devices, balancing the need for large numbers of relatively homogeneous devices with an embrace of heterogeneous, and potentially breakthrough, new kinds of hardware devices

Research in this area requires highly programmable devices, ranging from virtual machines to NetFPGA. Vendors are unlikely to support the wide range of programmable devices needed by researchers. In particular, current switch hardware is limited in programmability, while software switches support greater flexibility but are too slow for production settings. More generally, virtual machines can provide an excellent range of programmability, but do not provide good stand-ins for what is feasible in high-performance hardware. It is thus desirable to create some type of path by which researchers can transition their ideas from software-only prototypes to prototypes that reflect real hardware capabilities.

- Researchers must be able to control-plane/data-plane functionality boundary tradeoffs in switching, storage, etc., with the ability to put less or more functionality in the data plane
- Support a pool of relatively homogeneous hardware virtual slices so that synthesis tools can explore the space of platform-specific optimizations.
- Provide significant hardware diversity to stress the synthesis task. This requirement is motivated by the ongoing challenges faced during the shift of SDI network switches from OpenFlow API to the more expressive P4 interface, and the expected shift onwards to IOT sensors etc. This shift has forced the community to revisit SDI synthesis frameworks, as a result of the newer synthesis frameworks we have redesigned SDI verification frameworks.
- Researchers must be also able to leverage ongoing hardware breakthroughs (most notably in photonic switches and new forms of high-speed, nonvolatile memory)

Since developing/deploying the necessary high-speed, programmable hardware is both expensive and may require time-consuming development, these factors make this hardware/deployment an ideal target for a shared testbed.

2. Community repositories for shared tools, configurations, workloads, etc.

SDI verification typically starts with a collection of workloads and configurations. The end goal is to provide a set of provable properties that the network adhere to. As a result, any deployment that targets enabling future research on SDI verification will need to:

- Provide a common repository of workloads and configurations that the community collects from real-world measurements or produces as benchmarks to test the range of verifications tasks (i.e. common a SPEC-like benchmark for verification)
- Provide resources to be able to execute the synthesized output from the input configuration with high fidelity and replay the associated workload.
- While formal verification is an important step for ensuring correct operation, there is still room for complementary advances in testing methodologies to capture issues that are beyond the state of the art in SDI verification. The testbed should provide a suite of a programmable workload generators that enables orthogonal testing techniques to create input that stresses the synthesized output.

SDXes in Practice

Participants: Yang Guo (chair), Nick Feamster (scribe), Russ Clark, Chip Elliott, Arpit Gupta, Hongxin Hu, Hyojoon Kim, Balachander Krishnamurthy. Padma Krishnaswamy, Tom Lehman, Joe Mambretti, David Reese, Jerry Sobieski, Haoyu Song, and David Stern.

This session interleaved lightning talks by early SDX pioneers (including both researchers and experimental R&E infrastructure operators) with highly animated discussion and questioning among all the participants. Many viewpoints were raised and debate was sometimes vigorous. That said, by the end of the session, areas of broad agreement had emerged about the importance of the worldwide movement to Software Defined Infrastructure, the key research goals in this area, and specific ideas for next steps for SDI and SDX research, which are documented in the recommendations for this section.

We are already starting the transition of today's R&E cyberinfrastructure to Software Defined Infrastructure. As we move from today's Software Defined Networking to the emerging vision of Software Defined Infrastructure (SDI), we will need to describe, program, trouble-shoot, and reason about systems that span many different kinds of component devices and subsystems. Chief among them are those that support computation (virtual machines and/or bare metal machines), storage, connectivity, and many types of sensors and actuators. In addition, services will likely be incorporated into these over-arching systems. As is the case today, each of these components (resources) will be owned by someone. Therefore the basic issue is of creating end-to-end systems from components provided from multiple administrative domains.

Research and Education (R&E) cyberinfrastructure is in the early stages of a major transformation being driven by the emergence of these SDI technologies. In the context of this discussion, SDI is assumed to cover a broad range of technologies including host and storage system virtualization, Software Defined Networking (SDN), Network Functions Virtualization (NFV), Network Service Chaining (NSC), and other derivative technologies and functions. The R&E cyberinfrastructure space includes wide area networks, regional networks, campus infrastructures, and special purpose computational, storage, and cloud systems.

One example from the scientific domain may help make this clear. Today, we can connect a very high-bandwidth telescope in Hawaii across multiple networks (administrative domains) to its destination at Maryland, where local computation and storage is used to extract key features from this data flow. The result stream is orders of magnitude smaller that the raw sensor stream. Wouldn't it make more sense to move the computation and storage to devices close to the telescope, and then the processed stream to Maryland? This approach thus leads to the conclusion that researchers in Maryland might like to temporarily use clusters of computers and storage from a Hawaiian administrative domain, much as we often do with multi-tenant clouds today. They might also want to include application-dependent processing all the way along the flow from Hawaii to Maryland, i.e., within a number of administrative domains across the country. Today this is possible, but requires many human interactions. With SDI, it should be fluid and fully governed by software, so that such systems can be assembled, used, and freed up upon demand, within a few seconds.

Although this is an example from the scientific arena, there are many similar use cases from other domains. For example, organizations that use virtual networks may wish to install their own (NFV) devices through the network, such as load balancers or intrusion detection systems. Agricultural prediction services might wish to harvest drone sensor feeds from many providers, and aggregate the results regionally, etc.

Learning from early R&E prototypes of Software Defined Exchanges. This session engaged a number of researchers, and operators of research infrastructure, who have been active in creating SDI prototypes, and particularly those who have been creating very early versions of Software Defined Exchanges (SDXes). These exchanges are "meet me" points for different SDI administrative domains, and serve at points at which multi-domain applications can request resources from multiple domains, and from an SDX itself, to help create and operate multi-domain slices.

A vigorous community debate is underway regarding what exactly are the defining features and capabilities of this new SDI based cyberinfrastructure and the proper role(s) of SDXes. One vision is a new cyberinfrastructure where

network, compute, storage, and science instrument resources are managed within an integrated and orchestrated software defined paradigm. A key objective here is to release the science domain application and workflow developers from having think about, and manage, these resources in a separate and isolated manner. As an example, let us consider a Hybrid Cloud example. In the current environment, a science application workflow agent may ask for compute and storage resources from their local cluster, from a research facility, and/or from a public cloud. In addition, they may want these interconnected in a manner where the proper “Science DMZ” firewall bypass and other network configurations are in place as needed by their specific applications requirements. This may require provisioning of layer 3/layer 2 network paths, instantiation of BGP peers and some other NFV functions to connect all of these distributed resources together in a manner which the applications can use.

In this scenario the application workflow agent is required to "orchestrate" amongst distributed resources to build the needed application service environment. One vision for an SDI enable cyberinfrastructure is that that this "orchestration" function should be a service that the cyberinfrastructure provides. This will eliminate the need for every application workflow agent to recreate this in their own unique way.

Much research needed in this space. There are many research areas that must be addressed before this can be realized in a scalable, flexible, and feature rich manner. There are many individual technologies emerging for individual network element, compute cluster, or storage system software defined control. However, the infrastructure and technology to allow orchestration across a multi-domain, multi-resource, multi-technology distributed ecosystem is in the very early conceptual phase. What are good abstractions for describing, programming, debugging, and reasoning about such multi-domain systems that incorporate a very wide range of devices and services? Today we program individual devices, but surely we need to move to a model of programming these systems as ensembles. How can we understand, reason about, and ensure privacy and security in such systems? And once we discover innovative and useful new ideas in software based implementations, how can we rapidly instantiate those ideas in new hardware devices for efficiency gains?

Some of the key transformative research areas and infrastructure requirements are:

Transformative research areas

1. **Formal models for resource description and SDI service definition**
   - Models to flexibly describe resources, e.g., storage, network, middlebox, and compute, and how they interact with one another.
   - Models to precisely define domains, data flows, and end-to-end services
   - Methods and APIs supported by models in virtualized environments

2. **Abstractions, languages, and software for expressing network-wide service policies and facilitating multi-resource, multi-domain service orchestration in autonomous operator environments.**
   - Orchestration architecture and open APIs
   - Scalable, dynamic multi-domain federation

3. **Programmable measurement and analysis for performance verification and troubleshooting.**
   - Scalable monitoring in both the data and control planes without adversely affecting performance
   - Localizing non-performing components in highly virtualized environments
   - Support for per-hop, high-resolution timing, and the ability to extract this information for specific traffic flows.

4. **Mechanisms for ensuring security and privacy of both personal information (e.g., in data traffic) and proprietary information (e.g., business relationships).**
   - Authorization and resource sharing policies over shared infrastructure
   - Privacy-preserving performance monitoring (collaborative troubleshooting and sharing of measurements in ways that do not reveal private information)

Infrastructure requirements

1. **Fast (i.e., 100 Gbps+) forwarding in programmable hardware switches.**

Fast forwarding is especially important in two contexts: a) when connecting high-bandwidth scientific instruments to the emerging SDI infrastructure, and b) at aggregations points, such as SDXes, where a number of flows are multiplexed and managed as a high-bandwidth ensemble. At present it is difficult to find working switches that are both deeply programmable and that correctly handle 100 Gbps+ forwarding rates.

2. **Community repositories of "known good" software stacks and tools, from the controllers to storage and software infrastructure.**

At present, many researchers and R&E operators are basically reinventing the wheel, often over and over again. It is difficult to understand which combinations of controllers, switches, and OpenStack versions work successfully with each other. Furthermore, this landscape is shifting very quickly (e.g. on a week by week basis) and so it is very hard to find a stable, fixed point at which to perform experiments or to operate infrastructure. While the pace of change may or may not slow down in coming years, one important mitigation would be to have common repositories of "known good" combinations of software stacks, tools, and hardware versions which can be shared among researchers and R&E infrastructure operators.

3. **Mechanisms for faster implementation and deployment of new features in hardware, including the possibility of prototyping new hardware.**

While most academic researchers will focus on creating experimental software artifacts to demonstrate their ideas (e.g. new forms of load balancers, new content distribution systems, etc.), many of these new ideas will only be feasible in practice if they can be respun into hardware. It would be extremely help if NSF could help create a pathway from software artifacts to hardware instantiations. One example might be by means of protocol-independent switch ASICs, to support rapid development of more flexible packet processing pipelines in switches. Another example might be to set up an organization that can help researchers code their ideas in ways that can be transitioned to hardware (e.g. via Simulink), then can perform that transition should the idea gain widespread interest. We are somewhat agnostic about the best way to bridge this gap, but overall it would be helpful if there were some form of path from software prototypes to their hardware equivalents.

Education requirements

1. **Education and training of next-generation of SDI researchers and engineers, who must understand not only network protocols, but also software development that encompasses distributed computation, storage, and switching.**

As SDI concepts become prevalent, they will call for entirely new ways of thinking (in researchers) and new skill sets in technicians and undergraduates. At present, one might be educated as a "networking person" or a "storage person" but we expect that these current divisions will first blur, and then become obsolete. It would be helpful if NSF could fund workshops on the future educational needs that will be driven by the rise of SDI, and perhaps then begin to fund development of curriculum materials.

Co-evolving Applications and Infrastructure

Participants: Andy Bavier (Princeton University and ON.LAB), David Du (University of Minnesota), Rudra Dutta (North Carolina State University), Rodrigo Fonseca (Brown University), Mario Gerla (UCLA), Madeleine Glick (University of Arizona), Raj Kettimuthu (Argonne National Laboratory and The University of Chicago), Orran Krieger (Boston University), Larry Landweber (University of Wisconsin/BBN), Bryan Larish (NSA), Yan Luo (University of Massachusetts Lowell), Ben Mack-Crane (Corsa Technology), Ibrahim Matta (Boston University), Nagi Rao (Oak Ridge National Laboratory), Glenn Ricart (US Ignite and U. Utah), and Zhi-Li Zhang (University of Minnesota).

This session started with a series of lightning talks given by Bryan Larish, Glenn Ricart, Nagi Rao, Rudra Dutta, Zhi-Li Zhang, and Mario Gerla. The lightning talks were followed by a discussion which identified five transformative research areas and five infrastructure requirements. The concept of slicing and its impact on SDX/SDI infrastructure and applications was part of all the issues that were discussed in this session.

Transformative research areas

1. Design of cooperative information sharing schemes between Apps and SDI resources

SDI has enormous potential for co-optimization of application and the ensemble of diverse resources that it employs. Fundamentally, however, SDN to date has just been so far a tool to empower the system administrator at a particular layer. In a virtualized cloud environment, for example, even if the underlay is controlled by SDN, and the overlay is controlled by SDN, there is no communication between them, and there is no empowering of the application to take advantage of this broad programmability.

It is clear that there is information to be exchanged between layers that can help the network configure itself with information coming from the application layer, in much shorter timescales than is possible today, due to the programmable nature of SDI. It is also clear that the applications can be better optimized, operated, and deployed, with information coming from the network. An early example of this is the concept of Participatory Networking (published in SIGCOMM 2013). Today, in the cloud, for example, there are examples of tenants performing onerous reverse engineering of the cloud network topology, to select a set of VMs that are close to each other in the datacenter. Were information about the topology visible to the tenants, such effort would be unnecessary.

SDI allows this exchange of information between the infrastructure and the applications to be much richer and agile, however, there are big challenges to be overcome. Whenever the interface between two layers is widened, there arises the possibility of coupling between the two, and thus the decrease in the ability of the two to evolve independently. It is crucial for the exposition of information to be done in a way to, as much as possible, preserve the isolation between the layers, maintain a separation of concerns, and minimize this coupling, while, as much as possible, providing gains in performance, responsiveness, scalability, or security, to name a few. One example of such type of interface is the declarative nature of relational calculus. We need to define high-level interfaces for applications to express requirements, which then get compiled/optimized/placed into the infrastructure.

Another challenge for this exchange of information to be present is to have a model that presents the right incentives to the different parties to share information across the application / SDI boundary. In the example above, of the reverse engineering of the cloud infrastructure, as long as there is a near monopoly in the cloud provisioning, or there are too high barriers to switch among providers, there is little incentive for a provider to allow much visibility into its network structure. Another provider, however, with equivalent service, and the addition of extra visibility, may use this transparency as a competitive advantage.

The design of these cross-layer interfaces will require people who understand both sides of a layer boundary, such that many details are hidden, and the right abstractions exposed. One prime example of this is the interface with optical communication devices, which have features that could be very valuable to even today's SDNs. The

abstractions that we have today of such optical devices and media are too naïve, and provide no room for joint optimization.

Thinking in even broader terms, one question is “what are the abstractions for the cyberinfrastructure of 5 /10 / 20 years out?” In the same way that today's defining network abstractions are packets, ASes, the well-defined data plane layers, and principles like end-to-end, what are the things we will need to standardize, so that we can interoperate? Slices, SDIs, SDXs? The current Internet architecture largely left out security and accountability, and prescribed little or nothing of the control plane. It is likely that these new abstractions will have to include concepts like trust and manageability.

2. Machine augmentation of humans in managing highly fluid SDI / App systems

As computing, networking, storage technologies converge, and grow increasingly more sophisticated, and increasingly automated into agile software, an often-overlooked vulnerability in these information systems gains in risk even faster: the management, administration, and operation of such systems. Research has focused on improving the reliability of networking devices, protocols, and systems. But at this time, these protocols are already more reliable than the human administrators and managers who configure and operate them - at least in the enterprise networks, which are by far more numerous than ISP networks. The threat posed by misconfiguration of computers or networking systems have always been recognized, and every year brings its own misconfiguration-related disasters, such as the BGP misconfiguration in network AS9121 (TTnet - the largest ISP in Turkey), Google users’ loss of service(in December 2012), the infamous June 2013 Ztomy DNS misconfiguration.

Networking research has not addressed this problem. It has been assumed, perhaps reasonably, that more automation will solve this problem, or reduce it to negligible proportions. But experience teaches us otherwise. As software becomes more automated, the human tasks will become fewer, but more complex and higher-impact if done erroneously. In ARPANET days, humans (experts) entered routes by hand; today, OSPF and BGP automate these, but humans configure those protocols, a much more demanding job. Recent surveys show that these are typically not highly-paid jobs, nor highly-trained. The recently popular vision of SDN-based systems as a unifying future architecture of network-enabled computing systems has been seen by some as essentially eliminating the configuration issue. While there may be eventual truth to this, it is also true that configuration will now simply move to a more abstract, and more powerful, plane. In the five-to-fifteen year timeframe, humans will remain a significant part of the configuration loop. Over the same timeframe, the scope of configuration will expand and change unrecognizably as SDIs come into their own. As the job of the underpaid, undertrained network operations employee changes from configuring by hand to writing controller app code that can execute thousands of time in the seconds taken for a human operator to realize there is an error, we should look forward to ever more spectacular failures.

Research is needed to address at least two broad categories. First, the human process of the administration and management of SDI cyber-infrastructure systems while driven by their applications must be studied, to develop analytical and predictive models for this interaction. It is expected that such research will require collaboration from sociologists, behavioral psychologists, and perhaps draw on previous research in Human-Computer Interaction. Second, the design of such App / SDI cybersystems can integrally include the knowledge gained from such models, and thus provide a level of oversight to the human administrators. Research is needed on how to architect and design such intelligent systems, and what approaches or combination of approaches is most effective - for example, is it more effective to provide a “are you sure want to commit that step” prompt, or simply require a higher level of authority to perform certain functions?

3. SDI expanded and co-designed to include diverse end systems

SDI technologies must be developed to encompass end systems ranging from small physical systems to large science instruments, from tablets to supercomputers and clusters. As one example, a highschool student may set up an experiment on-demand to explore real-time data feeds from freshwater ponds around the world. As a larger, more complex example, tomorrow’s Large Hadron Collider (LHC) systems may be created and manipulated in a rich, highly fluid, and very resilient manner via Software Defined Infrastructure.

Complex data flows may be setup on demand or in-advance to encompass a variety of components. We can be sure that nearly all data flows will involve a combination of computation, storage, and communication

(networking), but a large number will also involve specialized sensors -- ranging from telescopes through gene sequencers.

This raises a number of key research challenges. How do we represent, control, and reason about such a wide variety of diverse end systems? How can we appropriately multiplex complex end systems (e.g. telescopes) so that they can serve as data sources for a large number of slices?

4. Interconnecting software-defined infrastructures

Software-defined infrastructure allows for infrastructure innovation at software rates hundreds or thousands of times faster than hardware can evolve. This is both the promise and the potential curse of software-defined infrastructure. While software-defined infrastructure can optimize within itself as much as desired, interconnecting two or more different software-defined infrastructures has the potential to become the Achilles heel of SDI if there is not also appropriate research on managing interconnections.

Can we develop appropriate standards so that the advantages of flexible software-defined infrastructure are not lost in converting to outdated standards at the interconnecting edges? One might assume that the abstraction to be preserved across software-defined infrastructures is the slice or flow. If so, interconnection of SDI infrastructures should heavily influence the properties of slices.

Can we match, preserve, emulate, or transform the properties of slices between two or more SDIs? We have significant experience in evolving the Internet using backward-compatible protocols while adding new capabilities and features. Can we extend these values of extensible protocols across SDI boundaries?

For example, to deliver end-to-end SDI, the interconnection will need to be cognizant of and match, preserve, emulate or transform:

- networking type for the slice (native OpenFlow, MPLS, GENI VLAN, GRE tunnel, SPB, etc.)
- service quality metrics
- control plane messages
- security encapsulations
- location / identifier mappings
- re-metering or re-timing flows
- emergency stop / rate limit
- economic issues (spending limits, time of day limits)
- notify / alert / log / report / bill / time-out

We also need to be able to provide for highly reliable assemblies of interconnected SDIs. This perhaps translates to highly reliable assemblies of slices, and that is also an important research topic.

Infrastructure requirements

1. Need for SDI testbeds supporting Real Applications and Real Users

As Orran Krieger said in his keynote, SDI/SDX testbeds "must include real user populations, and they have to have buy-in." That means real users running real applications. The first place for such testbeds has historically been university campuses, and we think this should be true for SDI/SDX as well. The first testbeds ought to be on-campus where there are knowledgeable people, innovative and disruptive users, and a bit of a moat to contain mistakes.

But once proven on campuses, the next real stop is the new hotbed of SDI/SDX innovation: cities, metropolitan areas, and other communities. The proliferation of the Internet of Things, machine-to-machine communication, personal digital assistants, and the density of urban areas all contribute to an inevitability that today's "take it to a distant cloud" architecture can no longer succeed. Instead, there will be more software-defined infrastructure closer to the edges (metro edges, wireless edges, and personal edges) to provide rapid response and frankly triage and containment for the deluge of new information that will be pouring forth. This is a kind of "locavore"

architecture in which the processing moves to the edge and consumes locally-generated data (just as a human locavore consumes locally-grown foods).

So community metropolitan areas will become the next hotbed for SDI/SDX and we need flexible Metropolitan testbeds. This is a direction compatible with US Ignite's initiative to deploy SDXes in metropolitan areas. The SDXes can both interconnect various SDIs and also sandbox them if they misbehave. With understanding civic leaders, carefully run SDI testbeds, emergency services available from SDXes, and appropriate prior research on the possible and likely impacts, we can learn a great deal by including opt-in portions of real user populations, and real applications. After all, gmail was officially in beta test when it was so valuable to real users it crossed the 100 million user threshold.

2. Shared test harnesses for repeatable research and performance assessment

Both researchers and R&E operators need a community repository of shared test harnesses that can be used to evaluate (and re-evaluate) behavior of Apps and SDI under various changes. This requirement includes both accurate measurement capabilities, and shared test harnesses. Once these capabilities are present, new technologies and new research ideas can be simply inserted into testbed and their performance figures can be determined in an objective and sharable fashion. For example, a new controller can be uploaded to the testbed harness and its impulse response will be generated; the controller will be tagged with this "standard" impulse response. This will make it far easier to perform "apples to apples" comparisons both for competing research ideas and for proposed modification to R&E infrastructure.

3. Testbed infrastructure for Education

As we move towards the new world in which Apps and Infrastructure co-evolve, and in which Apps can directly manage their underlying infrastructure resources -- growing and morphing the infrastructure in response to the App's current needs -- we will need to educate a new generation of students. This will require that some testbed infrastructure be devoted to education, and new curriculum materials be created and shared.

Connecting Across Domains

Participants: Tilman Wolf (chair), Ilya Baldin, Mark Berman, Marco Canini, Alvaro Cardenas, Xenofontas Dimitropoulos, James Griffioen, Cees de Laat, Raj Jain,Inder Monga, John Moore, Byrav Ramamurthy, Munindar Singh, Tim Talty, Malathi Veeraraghavan, John Wroclawski, and S. J. Ben Yoo.

This session considered research issues that will arise in multi-domain systems, i.e., those that span multiple SDI regimes, each of which is owned and operated by its own administrative entity. These entities might be fully cooperative with each other, partially cooperative, or indeed competitive. Clearly each entity will retain full ownership of its own SDI, and the means by which its resources will be lent (leased) out give rise to a number of questions of policy, markets, etc. In addition, it is unlikely that all domains will share exactly the same technologies (hardware, software) so questions of bridging or gateways also arise.

Ilya, John, MV, Mark, and Ben gave introductory lightning talks. The attendees then wrote down their thoughts about connecting across domains in the shared document. We discussed these thoughts and identified the three transformative research ideas and two infrastructure requirements below. We also noted that the following cross-cutting issues are important to consider: scalability, resilience, security, mobility.

Transformative research areas

1. **System-level SDI architecture (federated, multi-domain, etc.) and complexity management (modularity, composability, coordination functions); inter-substrate management**

To date, most SDI research has focused on systems that are a) relatively small in scale, and b) aimed primarily at the creation of rich functionality within a single administrative and management domain. As attention shifts to national and international-scale, decentralized, multi-domain deployments, new research questions become critically important. These new questions are concerned not with basic SDI function, but instead focus on *harnessing, controlling* and *managing* the potentially extreme system-level complexity created by today's rapidly expanding SDI functional capabilities. Example areas of concern might include:

- Defining and developing appropriate structuring principles, system modularities, and architectural approaches for large-scale software-defined infrastructures and systems;
- Approaches and techniques for designing and specifying functionally rich system-level and interdomain interfaces at appropriate levels of semantic specificity and abstraction;
- Development of scalable approaches to functional composition and coordination across decentralized, federated management domains within an overall SDI system;
- Development of metrics and evaluation approaches for system-level resilience, robustness, and performance properties in a large-scale, composed SDI system prior to full-scale deployment.

Summarizing, the overarching goal of this research area is to ensure, to the best of our ability, that SDI infrastructures and systems can develop over time to a level of scale, impact, and richness not yet imagined by today's designers, by creating the key structural principles, skeletons, and intellectual frameworks necessary to enable and facilitate this qualitative transformation from local to global perspective.

2. **Governance, privacy, market-based mechanism / business relationships / incentives, identity; application control; balance (tussle) between network provider control and application provider control**

Perhaps the most challenging aspect of multi-domain networking is to make things attractive for application providers and network providers. The challenge then is how do we balance the differences in the needs and the interests of various participants? What will be the incentives for each participants? Can there be a market-based negotiation mechanism to establish business relationships? The research topics can include:

- Governance of future SDIs: how can the SDI resources be coordinated for, for instance, high-throughput end-to-end services? How do we orchestrate them involving multi ASes and various applications?

- Business relationships and incentives: the symbiotic relationships between the application providers and the network providers need to converge towards business relationship. What will be the mechanisms for such convergence and what will be the incentives for both? Can market-driven brokers help?

- Multi-domain SDIs are challenging because of security policies. For instance, domain managers are reluctant to provide domain specific information like internal topologies and traffic matrix, but they can receive better inter-domain networking services (e.g. end-to-end throughput across multi-domains) if such information is shared. Tradeoff and balanced optimal operation are important.

- How will the SDI evolve and how will the negotiation points change over time? How will market-driven evolution lead towards a better future SDI?

- Application providers should be able to convey their desired policies to various resource providers (ISPs, IXPs, Cloud Service Providers, CDNs). Networking alone is not sufficient to ensure the end-to-end QoS/QoE. Heterogeneity of APIs from different service providers will be a norm.

3. Information models and policies, enforcement

Scalable efficient resource management (allocation, provisioning, FCAPS) in a multi-domain SDI environment will require capturing the heterogeneity of network element types, links and paths and their properties or metadata. We cannot think about connecting multiple SDI domains using automated software without a clear information model through which topology, resources, capabilities etc. can be expressed and shared. The data about the connectivity, configuration and the state of the network will be collected, maintained and exchanged by multiple agents within this environment, ultimately making the problem of managing this distributed information a yet another kind of 'big-data' problem, where data movement and granularities of its disclosure are governed by provider business policies.

Yet, there do not seem to be well accepted and expressive mechanisms, language, abstractions to represent either the resource information or service policies that can be operate at the level of multi-domain environment. There is an explicit a need for rigorously developed information models and extensible representation mechanisms that allow describing heterogeneous networked resources. The information model should support multiple views of varying levels of detail, and support virtual views of the network infrastructure that, when merged with monitoring data, lend themselves to large-scale analytics in support of efficient operation of the multi-domain SDI environment in the interests of individual providers, tenants/users. In addition, policy languages and enforcement mechanisms are needed to allow for efficient sharing of this information between various entities. This will be particularly important when introducing market-based mechanisms, like brokering, into the multi-domain SDI environment.

Infrastructure requirements

1. Widely-distributed SDI, of which portions are owned and operated by different administrative entities to give rise to real-world multi-domain issues. Other SDI regions might be organized so that they could set up as "mock multi-domain" infrastructure in order to try out new ideas in multi-domain architectures, policy mechanisms, etc.

The infrastructure that supports this research agenda needs to be viewed as a system that encompasses campuses, regional networks, scientific instruments, and international partners. Interconnection points and international circuits require common, flexible policy regimes: infrastructure investments that carry restrictive terms of use should be avoided. Robust governance of the infrastructure that can take the systems view and effectively operate internationally will be key.

To maximize the potential range of software defined infrastructure capabilities, it is important to stress the limits of our ability to describe, allocate, virtualize, interconnect and interoperate among different cyberinfrastructure technologies crossing multiple administrative domains.

Fortunately, the natural arc of university research projects leads in this direction, as different institutions stand up diverse resources to address the specialized research of their respective faculties. Universities and their research sponsors are already actively fielding common computing, networking, and storage infrastructure, as well as specialized scientific instruments, cyberphysical systems, sensing, and high-performance computing resources. However, sharing these resources across a broad scientific community is not always a priority. Institutions should be strongly encouraged find ways to slice these resources and make them available to interested researchers beyond their own campuses, subject to appropriate policy constraints.

The platform must be thought of as part of a continuum with production services that are vital to support discipline science with high capacity, reliable services. Ideas proven to be useful and effective should have a streamlined path into production products. The challenge is to do so without creating dependencies on mainline vendor products whose feature sets are driven by considerations other than research needs. Developing R&D partnerships with vendors working in this space is key.

2. Open SDXes as connection points, open repositories of reusable software

Testbeds need to be built that will allow researchers to experience, experiment with, and test multi-domain features. One of the best approaches to inter-connect testbeds is to leverage SDX's as interconnection points. Direct connection of testbeds makes it less flexible and also, might break the security model as data and control planes need to get connected. Heterogeneous testbeds might need to connect at various layers of the data plane.

In addition to the above, each testbed could have their own mechanism for authentication, authorization, control and an environment to run experiments and collect results. When interconnecting testbeds, it will be very helpful to have access to reusable software components that deal with common testbed features. GENI is one example of such a project. As people build newer SDI testbeds, this sharing might become hard. Through collaboration, the academic community can repurpose GENI software and build the components in a format that is easily reusable and can enable multiple companies to become part of a GENI-like project.

The Future of Programmable Network Hardware

Participants: Marco Canini, Yang Guo, Yan Luo, Ben Mack-Crane, Dan Kilper, Hyojoon Kim, Balachander Krishnamurthy, Bryan Larish, Jerry Sobieski, Haoyu Song, and David Stern.

We first had three lightning talks by Ben, Dan, and Haoyu. We then proceeded to a discussion around the problems raised in the talks as well as other thoughts from the attendees. We wrote down our thoughts regarding the underlying challenges in the shared document. We identified what we believe are three main research areas in the general space of programmable network hardware. Because the topic is arguably already hot, we focused for the 5-10 years horizon, recognizing the issue of programming across devices as a larger problem than what current works are undertaking. Important new technologies such as photonics also need to be embraced. We also identified two sets of main infrastructure needs to catalyze research along these themes.

Transformative research areas

1. **Programmability across heterogeneous devices and technologies, and the realization of large scale programmable 'super systems' such as programmable cities**

Smart cities and other large scale enterprises will rely on a wide range of networked devices and associated heterogeneous network technologies including next-generation cellular technology, software defined radio, visible light communications, fiber optic networks and wavelength switching, packet, circuit and flow switches. They will also rely on the incredible advances being made in today's storage technologies, and even more so the emerging technologies that we can foresee in the next 5-10 years.

Revolutionary advances in these devices should be explicitly embraced, as greater integration at the device, subsystem, and system levels will enable unprecedented scales of hardware deployment. Realizing seamless programmability across all of this hardware is a major research challenge. Heterogeneous hardware will need to support abstractions necessary to work together and to become a part of larger scale but unified systems such as a "programmable city." Research will be needed into which details, controls, and monitoring information should be shared within the abstraction models, how they fit into standard templates, and the modalities of how this information communicated (frequency of updates, etc.). Programmable hardware will need to enable a level of trust that the large scale systems can utilize them without causing disruption or harm across the system.

2. **Safety and verification of programmable devices**

In the next few years an increasing variety of fully programmable switching devices will become available. Research topics will emerge regarding how to most effectively program the facets of datapath behavior that have traditionally been bound (and thus fixed) in ASSP designs. These facets include header formats, packet processing functions, and pipeline structure (sometimes expressed as network layering).

Related kinds of issues arise in multi-tenant cloud systems, in which we need to employ trusted software in order to reliably provision virtual machines, enforce isolation, etc. Again the questions of safety and verification of programmable devices are extremely important.

As many more SDI devices (resources) become programmable it is important to develop an understanding of the operational impact of the programs running these underlying devices (switches, computers, storage systems, etc.). Some of the questions that will need to be answered are:

- What features are needed, and perhaps as importantly <u>not</u> needed, in a device programming language?
- How can one validate that a device program meets a set of behavioral requirements?
- How can one validate that a device program will operate correctly in the context of an existing network?
- How can one validate the consistency of a set of devices programs (e.g., programs for different switch roles in a network architecture) and ensure the edge-to-edge behavior of the network design satisfies a set of requirements.

- How can one understand and control (or validate) the impact of a change to a device program? For example, if a change is to be made in an active switch to install a custom monitor, how can one ensure the change will only perform a monitoring function (extracting information from the switch) and not impact forwarding or or other critical switch functions? (This can be seen as answering the question "How can we know a datapath program change is safe?" for some definition of "safe".)
- What mechanisms can be used to install device program changes that protect active applications that are relying on these devices?

3. "Open chip" hardware for SDI experimentation

Today there are many different chip-level hardware architectures for supporting SDI, including commodity CPUs, GPUs, FPGAs, network processors, and so forth. At present, these architectures are presented as a "given" to researchers who are exploring software architectures for SDI, applications, etc. Can we move to something more akin to an "open chip" architecture, in which researchers can instantiate hardware of their own ideas, which can then be tried in through simulations, power modeling, and ultimately use in experimental infrastructures via benchmark applications (e.g. NFV)?

Infrastructure requirements

1. Stable, large-scale SDI environment that can also interoperate with smaller, more experimental regions of advanced technology (e.g. in photonics, storage, etc.)

To fully utilize these new programmable features, researchers need a realistic, programmable infrastructure. Making this infrastructure realistic requires large scale both in terms of size (e.g., in a smart-city-scale deployment) and users interacting with the infrastructure. It also requires programmable hardware across multiple technology platforms (e.g., sensors, IoT, wireless, optical, switching). To ensure researchers have access to the latest technology, partnerships needs to be formed with hardware vendors to get early access to products. Partnerships should also be formed with service providers (e.g., wireless) to ensure the programmable infrastructure is deployed in all environments without causing regulatory or legal issues.

Many regions of this large-scale SDI environment will need to be stable, so that researchers can use the infrastructure without worrying about its reliability. However, some smaller regions should explicitly embrace new technologies that appear to provide major new capabilities. Advanced photonic networks come to mind, as do experiments with next-generation storage systems. A similar approach might also work for new kinds of advanced sensors. For example, one or more campuses or cities might be provisioned with an advanced photonic network and many sensors, in order to provide experimenters with an early view into the kinds of technologies that we expect to emerge in coming years.

2. An SDI testbed that permits experimental technologies and applications to coexist with more mature (production-level) applications and services

As our networks become more infused with a full range of e-Infrastructure (compute, storage, etc) and these facilities become ubiquitous and part of an emerging new smart environment, the ability to develop and test at scale with real users and real world physical environments become critical to refining and delivering mature capabilities. Thus we need hardware and software environment that allow experimental technologies to co-exist side by side with more mature or production services and applications. There needs to be a formal model that recognizes this dichotomy, and allows applications and services in a wide range of experimental maturity to be easily and "safely" deployed.

For example, a "virtual application network environment" capability that is a fundamental architectural part of smart cities or the IoT (and/or other scenarios) that always provides a separation of these application networks so that they do not interfere with one another or interact with one another in explicit and deterministic fashion is needed. This might allow a new application to see and manipulate certain sensors or sensor data, but not sensors or data that might interfere with other existing applications or policy (a related aspect is to develop a predictive verifiability that applications can *only* access or modify the intended data in the intended fashion - as a mistake could be quite damaging in the real world.) The virtualization model therefore not only need exist in the hardware

or in the control framework, but perhaps as part of the data or information model as well - i.e. virtualized data objects. Another example was the passive access to even just monitoring data, when scaled up to thousands or millions of sensors, could create data transport or assimilation requirements that could impact other co-existing services. So the entire application must be considered - not just the mote or smart devices that host the active portions of the application.

Thus a formal model for building and embedding applications into the smart environment in a manner that insures different applications interact with one another only as intended is needed. The discussion used smart cities as an example, where the community e-infrastructure was ubiquitous and amorphous but could be viewed as a giant intelligent device but programmed in an intuitive manner. Specific hardware should support such partitioning of the component for delegating and insulating.isolating applications elements.

3. Computer science testbed that incorporates research level photonics which would also serve as a basis and focal point for education and bringing communities together

The computer science community is currently widely separated from the research photonics community, and thus there are very few interactions that could take advantage of the combination of computer science with advanced photonics. This is truly a missed opportunity, as can be seen from contrast with the extremely fruitful interactions of computer scientists with RF radio communities, which has given rise to remarkable innovations in software defined radios, using WiFi systems for geo-location or other non-communication uses, etc. We believe that similar kinds of innovation could arise from bringing the computer science communities together with those of advanced photonics.

SDI as a Marketplace

Participants: Rodrigo Fonseca (Chair), Jim Griffioen, Orran Krieger, Cees de Laat, John Moore, Bruce Patterson, Glenn Ricart, Munindar P. Singh, Tilman Wolf, John Wroclawski, and S.J.Ben Yoo.

We began with lightning talks by Bruce Patterson, Cees de Laat, James Griffioen, Munindar Singh, and Tilman Wolf, then engaged in a very broad and interesting discussion on several aspects of the markets that different instantiations of SDI enable. Concretely, we talked about three types of SDIs: first, edge-based SDI in the form of programmable boxes deployed close to homes, where end-users can select multiple providers and services, and create custom networks among devices and users. The second SDI we talked about was SDX, and the third was cloud environments.

In these contexts, we talked about the economic models enabled by the fine granularity and timescales of compositions of services. One big question was whether the economic mechanisms (negotiations, contracts, exchange of money) have to be tied to or decoupled from the technical mechanisms. The consensus is that they should evolve independently, and the technical mechanisms should enable both traditional and novel economic models.

In a marketplace there will be multiple agents that are potentially self-interested and even distrustful, so one topic is how to get them to cooperate, and share information that is mutually beneficial, through incentives and/or through social norms, at different levels of enforcement. SDIs will certainly increase the opportunity for composing services at different levels. We discussed some of the new problems that operating on multiple pieces of infrastructure you don't own, regardless of what the economic models and the norms (contracts, enforcements) are.

Lastly, in terms of infrastructure, we discussed the need of testing these models with real users, real money. When not possible, tap into research on agents and market simulations to understand implications of systems at potentially large scale.

Transformative research areas

1. Investigating new economic models and mechanisms for (multi-domain) resource use

Orran Krieger, in his plenary talk, provided an interesting example of a realtime medical imaging application that might acquire thousands of cloud computers, use them for a second or two, and then release them; such intense burst of resource usage are not well supported in today's clouds. To broaden this to tomorrow's SDI, how can we set up technical and economic mechanisms for very fast "burst" usage of resources, with fine granularity, across a wide area? On another front, there are open issues in whether and how economic mechanisms should be tied to technical mechanisms; for example, are the mechanisms used to acquire / lease resources intimately tied to some kind of billing mechanism, or are they decoupled enough so that many different economic mechanisms could be used with a single technical mechanism (and vice versa)? As part of this research, we expect that even the mechanisms for payment should be negotiable; this becomes both important and complex as slices extend across multiple administrative domains, which may in turn have radically different economic / monetary policies.

2. Incentives for Cooperation / Governance of Autonomous Entities

For a vibrant and live ecosystem of SDI entities to be delivered by service providers used by applications we need an architecture that describes the relationships amongst all entities on both infrastructure and social level. Within such architecture different trust models must be explored to find the ones that scale and work for the communities that are served by the SDI providers, and that scale for those providers. Such models help the SDIs to come up with the desired marketplaces where services can be build upon.

- Investigating how Service Provider Group concepts can be applied to a collaboration of service provider organizations, where: each provider can offer slivers that can become part of a slice stretching across

multiple autonomous aggregate managers, such slivers have a common notion of service quality requirements, are the key subjects this research contribution.

- Crucial to such an architecture is an implementation-independent standard of correctness that we can use to determine the compliance of the policies adopted by various autonomous parties. Formal representations drawn from law and contracts, especially computational models of norms, are promising in that they (1) are conceptually close to concerns of business relationships and collaboration; (2) support formal reasoning; (3) provide a basis for key performance indicators, which in turn are (4) bases for administrative tasks such as monitoring. Research challenges include developing (1) sophisticated models that are *externally valid*, meaning can adequately capture stakeholder requirements and are comprehensible to stakeholders; developing decision procedures for verifying these models with respect to criteria such as liveness and resilience; (2) formal techniques supporting design-time verification; and (3) techniques for verifying specific actions and producing recommended actions that would satisfy a given model.

3. Exploring the marketplace implications of "SDI to the Edge" and "premise SDXes"

Slicing to the premise edge (e.g. a household) clearly provides new layers for innovation - considerably more than Internet as conceived today. New models enabled by programmable infrastructure at the edge could include both new user models, such as:

- Layer 2 point to point or point to multipoint private networks
- Improved security through new lower layer control and visibility at edge
- New dynamic functionality again through lower layer control and visibility consumable at edge

and new provider models, such as:

- Transport utilities: facilitates the separation of services (layer 3 and above) from transport (layer 2 and below) by providing a demarc for each stakeholder (network operator, service provider and subscriber) at the premise
- Cloudlet at the home: Accommodates the virtualization of routers, switches, wireless access points, LTE and middleboxes. Middleboxes could include firewalls, NAT devices, DNS, DHCP, intrusion detection systems, IoT devices or other unexpected devices.
- Carrier transparency to the edges: Provides for OAM functionality at each stakeholder's demarc. Testing such as RFC2544, ITU-T Y.1731, IEEE 802.1ag, etc.
- Smart grid enabler: Mesh Zigbee or 6lowpan could be privately and securely connected through edge slicing to provide private network and ownership to and in the premise for power, water, gas, sewer at every address. For municipal, county or others, similar capabilities for traffic at street lights, cameras, public safety (FirstNet) or any other entity requiring always on virtualized network ownership with addressing and prioritization controls.

As an example: Could next generation Public Safety telecommunications services be operated as slice aware application(s) to the premise? A provider with agreement from the end user or subscriber would be able to apply Class of Service (COS) and/or specific paths, assets to individual services and/or applications. Research in this area could provide a path to answer current Net Neutrality policy question and allow for the introduction of additional functionality to the current Internet. Does providing an end user the ability to apply COS to services violate or support Net Neutrality? Do we expect the Internet of tomorrow to provide these types of end user controls?

Depending on specific SDX definitions, an argument could be made that homeowner of the future will own and operate a premise SDX at the home. In some sense a premise exchange already exists at the home within the typical WiFi router. With the coming switch to globally unique addressing where will traffic decisions (filtering, blocking or routing) be made? Will the end user want to make route decisions in the future (maybe based on security, performance or price), and if so will they be provided with the vision and toolset to do so? Does IoT in some sense create a distributed SDI and if so where and what kind of exchanges will be required?

While these questions can really only be answered in some future where programmability is available to the network edge user in a consumable manner, it clearly argues for a 'premise 'SDX' as the replacement for today's WiFi router. Note: correct SDI/SDX implementation provides for programmability redirection (ie: orchestration or 'profile preferences' management) could be outsourced by the premise owner (as a service or to a trusted representative).

4. Service interoperability and composition at SDXes

Service interoperability between SDI slices would appear to be the major function of an SDX. Between similar types of SDIs, it may simply be stitching slice flows together. Where there are differences in the slice flow mechanics (e.g., MPLS to OpenFlow), some kind of intermediate gateway function will need to be performed where each side sees its own abstraction although it's different from the abstraction on the other side. Research is needed to determine the best way to create and manage these gateways and to determine which kinds of performance and economic penalties are inherent due to the entropic penalty of the mismatch and which can be optimized or standardized.

But SDX transformations don't have to be least-common-denominator connectivity. They can add value and function. The SDX can be looked upon as having intrinsic value (just like NFV functions) by adding capabilities to slice mating points including logging, fault isolation, billing, performance measurement and tracking, and so forth. But it needn't stop there. Other values can be added (e.g., translation, encryption, transcoding, re-timing, multicasting, and sending multiple copies over diverse paths for reliability). This then becomes a service composition and chaining issue for slices. All of these are fertile research opportunities in network-based composition, assembly, and abstraction.

Should economic models follow these technical mechanisms? Or should they preserve future flexibility by only measuring higher-level functions? Another area demanding research is dealing with mistrust. Can or should SDXes become trust boundaries? Should SDIs inherently distrust other SDIs (as we have learned to do with protocols)? Should SDIs distrust the SDXes with whom they deal? What kinds of trust can be assured and at what levels with what mechanisms?

Finally, what are the norms for acceptable behavior between SDIs and SDXes and other SDIs?

Infrastructure requirements

1. SDXs deployed at different scales (providers and users)

We've been thinking about SDXs as analogous to Internet Exchange Points that connect traffic aggregators, content providers and other large sources of traffic. However, the larger SDI space will likely contain infrastructure that is highly distributed, with smaller-scale devices situated closer to the user. The research questions addressed in this session such as service composition and interoperability, cooperation and governance between participating entities and exploration of various economic models require testbeds that operate at multiple scales.

An excellent example of smaller-scale infrastructure was presented by Bruce Patterson of the City of Ammon, ID. His team has developed hardware that is installed at a residence or SMB that provides programmable selection of ISPs and other services, as well as the ability to cooperatively deploy services between end users. There seems to be an opportunity to work collaboratively on these research issues between the existing larger scale SDX testbeds and efforts operating at the municipality or metro area scale such as US Ignite and Smart Cities initiatives.

In fact, the entire notion of "edge" or SDI might turn out to be somewhat recursive with SDIs within other SDIs but at different scales. This notion is similar to software-defined instruction execution for certain instructions on certain processors. There may well be SDIs at multiple levels, and we don't yet understand the interactions this is likely to cause.

For example, it seems reasonable that in addition to Metropolitan SDIs serving the gazillion smart devices in its Metropolitan area, there will also be SDIs at every point where wired feeds wireless. The SDI at this second edge is

needed for wireless mobility and other services anyway, so why not do some of the applications and attendant infrastructure work closer to the wireless devices and improve scalability? And smartphones are already acting as a personal edge, connecting to WiFi/cellular on one hand and low-energy Bluetooth on the other. They relay, for example, fitbit data to the next higher level edge while aggregating it for the user. Is this an appropriate architecture? What are the right abstractions for somewhat recursive edges? How should processing find the correct level of edge?

How should SDIs at different scales operate together to deliver services efficiently? What are the economic models that will apply hierarchically? Or are hierarchies the wrong abstraction? How can economics play their usual role of optimization in multi-scale architectures? How can or should real users understand or even care about multi-scale applications? How can we incent enough real multi-scale applications to be written so that we can research these questions using real users and real applications?

2. Agents and simulations

Given the natural tension that exists between infrastructure being a testbed and having real users dealing with real money, our field could use lessons and resources from other communities that deal with similar issues. One prime example of this is auction research, which customarily uses agent competition and simulations to model users of these markets and examine different strategies, scenarios, market rules. ACM EC is an example of a longstanding series of conferences that include work along these lines. Not only could funding for simulation environments (analogous to what ns2/3 has been for pure networking research) be beneficial here, but also funding for agent competitions in this area could prove valuable to the community.

Interconnecting Across Different Technologies

Participants: Mehrdad Moradi (University of Michigan), Mark Berman (scribe, BBN), Alvaro Cardenas, James Chacko (Drexel University), Russ Clark (Georgia Tech), Rudra Dutta (North Carolina State University), Chip Elliott (BBN), Madeleine Glick (University of Arizona), Hongxin Hu (Clemson)Tim Talty (GM), Joe Touch (ISI), Anduo Wang (Temple), KC Wang (Clemson University).

This breakout session focused on the quickly expanding diversity of programmable resources. Even while noting that our discussion was perhaps artificially focused in specific areas because of the expertise of participants, we quickly observed that these programmable resources extend well beyond what would traditionally constitute "the network." We discussed the potential of several different sorts of technology, as described by participants. Storage systems and even individual storage devices are becoming more capable computation environments, but their capabilities are not easily exposed to most application developers. Similarly, optical networks are very exciting for their high bandwidth capability, but their programming paradigm is different from how OpenFlow, say, has been adopted. Our cars are quickly becoming very sophisticated sensor, networking, processing, and storage "clouds" on wheels, opening up many new opportunities, but with new constraints arising from safety, privacy, and proprietary technology concerns, among others.

We discussed a number of highly evocative research questions that arise in this context. What role will massive "data torrents" play in future SDXes, particularly when this data comes from many different owners? How will SDXes play as convergence points for storage, in addition to networking and computation? How will "data peering" work? Are there opportunities to chain novel FPGA or other hardware-supported services into a NFV-like service chain in within an SDX? Can we experiment with novel photonic interconnects within an SDX? What role will cars play in 5-10 years, given that they will contain very high-bandwidth, real-time sensing (e.g. lidar and video)? Who owns this data and how can it be shared while maintaining privacy? How will taskable drones fit into tomorrow's SDI?

There's tremendous potential in offering access to heterogeneous groupings of these diverse technologies. However, programming each of these can require special expertise. Research efforts to make programmability more accessible and to build applications and services on these new programming capabilities are a highly promising area. We recommend pursuing these advances in the context of multiple experimental SDXs, each with diverse types of programmable cyber resources. It is important to note that the working group recognized our own limitations growing from our personal research interests. The group's larger intent is to emphasize the importance of working across diverse types of infrastructure resources and their associated data and services. We recommend this goal as a major thrust for software defined infrastructure research.

Transformative research areas

1. **We foresee rapid introduction of large amounts of distributed storage, computation, and an extremely wide variety of sensors and actuators (e.g. ranging from household IoT devices through automobiles to city-scale systems). How will these devices be incorporated in Software Defined Infrastructure? How will we describe, task, program, manage, debug, and reason about distributed software systems that incorporate such a wide variety of devices?**

It is readily apparent that we are now entering an era of cheap, ubiquitous sensing, computation, and storage. Although the term "Internet of Things" covers a huge variety of technologies and systems, two fundamental technology drivers will continue (and perhaps accelerate) for the foreseeable future. First, a variety of devices will have substantial computing power and local storage capabilities. Storage is of particular interest as it is undergoing a technological revolution at present (e.g. flash and memristors), and may also be moving from relatively closed storage system architectures to a more open, flexible "software defined storage" approach. Cheap, massive, persistent storage will become the norm in most devices. Second, sensors are rapidly becoming ubiquitous. Taken together, these trends indicate that our 5-10 year future will be filled with devices and systems

that contain substantial local computation, storage, and sensing capabilities. These devices can be mobile or stationary. They are continuously monitoring our environment and producing data.

Automobiles are a particularly interesting case in point, as they will soon have an extraordinary suite of high-bandwidth sensors (needed for semi-autonomous and autonomous driving). Combined with the envisioned next-generation wireless systems ("5G"), these cars will become roving, very-high capacity distributed sensors. One can imagine "tasking" automobile sensors for many uses in addition to supporting autonomy. For example, during snowfalls, one might collect automotive data (e.g. from tractional control) to determine which street locations are snowy or icy, and hence need plowing or salt. Automobiles may also perform near-continuous, fine-grain 3D measurements of the environment (e.g. via lidar), which when aggregated will provide realtime inputs to 3D models of our cities or countryside. They can also provide very fine-grain measurements of local weather conditions, which could greatly improve local weather prediction. But fundamental questions arise with such scenarios: who can task an automobile? what kinds of controls does the automobile's owner impose? how can we arbitrate potential competing demands for tasking of such sensors?

In short, our future SDI should have the capability of incorporating a wide range of static and mobile devices, permitting (controlled) tasking of their sensors and actuators, and managing, delivering, and preserving the copious amounts of data which will permeate this highly distributed environment.

2. How will we do data peering?

We already live in a world in which we are "drowning in data," but if we project forward 5-10 years, we can easily envision a world in which massive amounts of data are continuously collected and may be shared for a variety of applications (e.g. as in the automotive examples above). Who owns this data? How can we merge ever-evolving sets of data, from many different owners, to obtain useful products? How do we preserve any form of privacy in such a world? What new security issues arise?

Today's Information Centric Networks (ICNs) are taking a first step towards grappling with such issues, but they do not yet adequately address what we might call "data peering" issues, i.e., the selective movement and use of data controlled by different administrative domains. At present, Software Defined Networking begun to grapple with "connectivity peering" issues; one keynote described the use of SDX's to provide such connectivity peering between today's ISPs. Looking out 5 to 10 years, however, data peering may be a much more important issue than connectivity peering. Consider the automotive sensing ideas described above; if each car is its own "data administrative domain" for its owner, we will need to "peer" data from many thousands of domains to obtain useful results, with the resultant products then going into numerous additional domains (city public works, traffic management, weather prediction, commercial uses, …) The same case can be made for data generated by drones with many different owners.

What mechanisms in Software Defined Infrastructure will allow data to be real-time delivered in the interested parties in a controlled manner? What security and privacy requirements arise, and how might they be addressed? What market systems might work for data peering? Do we need to expand our notion of SDX from a neutral "connectivity peering point" to something more like a "data peering point"?

3. SDI operating system - raise the level of abstraction of programming multiple resource types

End system heterogeneity, coordinated sharing, and support for user/application requirements is currently supported by the concept of an operating system (OS). OSes abstract applications from low-level machine details, coordinate sharing, and ensure that concurrent sharing avoids interference. As we progress with more complex SDI constellations and more heterogeneous network capabilities (including in-network storage and processing), we start to need a "network operating system" (NOS).

This NOS could provide not only these basic features, but like an OS, this many also involve transitioning end-system middleware into the network as well. Middleware is a virtual service that includes persistent state (otherwise it would be just a library), which incorporates aspects of the end system and OS across multiple machines but is not supported in either. In SDI, that middleware can become part of the NOS that supports stateful coordination and services within the network components. The NOS is assisted by advanced approaches to "Models for programming and verifying SDI" (see Breakout Section 1), so that both together can help translate

user/operator intent into a distributed set of SDI configuration and services. The NOS can also interact with end system OSes to help coordinate end system intent and capability with SDI capability.

Infrastructure requirements

1. New paradigms for low friction approach to interconnecting technologies

Interconnecting the potential components of the SDI is doable today but with significant efforts. Crossing the boundaries of different technologies, e.g., between wired and wireless network domains, between sensor networks and their backhaul networks, between mobile networks and the Internet, or even between networks of the same technology but different administrative domains, their interconnectivity often requires human configuration, and visibility often does not transcend the boundary. To truly enables programmable control of SDI, these technologies need to be visible (perhaps via suitable abstractions instead of raw topology or config) across domains as a prerequisite for programmability.

Friction also results from the disjoint associate process to each different network domains, and the resulting overhead poses severe performance limitation for such SDI to support high performance applications that need to flexibly move/relocate across network domains. One solution to reduce the friction across SDI is to enable the least overhead connectivity options for each technology to, ideally automatically, achieve connectivity, while treating functions such as authentication, admission, and security as customizable services that can be instantiated on demand in the network data path. Network Function Virtualization (NFV) research well suits this need. As a result, we envision an ideal SDI to have robust, minimal overhead connectivity throughout, while end-to-end applications would customize the data plane on demand through SDN-enabled service chaining, inserting either physical or virtual services.

2. Need multiple SDXs, with significant data storage and wireless connectivity to emerging real-time data sources (IOT, sensors, cars, drones)

Software Defined Exchanges (SDX's) serve as the "meet me" points for multiple administrative domains within Software Defined Infrastructure. We envision that these meet-me points will provide connectivity peering services (as in today's Internet) but will also provide meet-me points for services such as large-scale content distribution, multi-domain clouds, etc. With the rise of large numbers of high-capacity sensors (e.g. automobiles and drones), they may also serve as meet-me points for many different data domains. As such, they will clearly require sufficient local computation, storage, and connectivity to support many different services simultaneously via a "multi-tenant" or sliced approach.

The architecture, and indeed even the basic requirements, for such future systems is still very much a research area. Hence we will need multiple SDXs to explore this new environment, so that different (and probably radically different) approaches can be explored by a number of researchers. In addition, some or all of these SDXs may be strategically located so that they can support continuous, high-bandwidth sensing from multiple sources, so that issues of data-peering may be explored. For example, some SDXs may be located in communities with automobiles connected via high-capacity, low-latency wireless links, so that their automotive sensing capabilities may be explored in a broader context as described above. We should not restrict our thinking to automobiles, however. For example, an SDX co-located near other high-capacity sensor/actuator systems (e.g. fleets of drones) can also provide a very interesting environment for such explorations.

3. Need at least one SDX tailored for experimentation with novel "data center" photonics interconnection technology within the SDX, and which is connected to multiple optical networks

There will be massive increases in the data being exchanged and stored in the network in the coming years. We know that the use of optics and photonics has significant potential to aid in the transmission of large amounts of data; however, there are challenges to incorporate the analog optical components into the current SDX framework. We would like to see interfaces that are agnostic to the specifics of the underlying technology for multiple reasons. Optical technology will change over the next 5-10 years and the interface should be able to take this into account. In addition.

DIfferent types of users will require different levels of exposure to the underlying technology. For users who are not experts, there should be an ability to establish light pipes without the user's specific demand (it does not have to be obvious or relevant to the user which physical media is being used). Similarly, there needs to be an infrastructure to enable transparent switching between wireless, wired and optical communication. The hardware across these three technologies are very different, eg. digital vs analog elements, and current technology in these areas do not sub set well across each other, although they achieve similar functionally. An abstraction to functionally target one implementation over the other would work well here. However, for specialists, it will be desirable to have direct exposure to and active control over the underlying technology to permit experimentation. These goals can be achieved through an experimental testbed or infrastructure that incorporates advanced photonic technology into an SDX setting.

End-to-end: SDI across the Wide Area

Participants: Ethan Katz-Bassett (chair), Aditya Akella, Byrav Ramamurthy, Fontas Dimitropoulos, Inder Monga, Srini Seshan, Andy Bavier, Arpit Gupta, David Reese, Ibrahim Matta, Ilya Baldin, Joe Mambretti, Malathi Veeraraghavan, Mike Zink, Nagi Rao, Raj Jain, Raj Kettimuthu, Rick McGeer, Theo Benson, Tom Lehman, Yan Luo, and Zhi-Li Zhang.

We discussed multiple use cases which could benefit from end-to-end software control. This included video delivery, science DMZ-style huge data and supercomputing, Internet paths with QOS for critical services, and application spaces such as IoT/healthcare/vehicular. In-depth discussions of these different domains led to common elements emerging. In particular, controlling routes alone is not enough: providing QoE/QoS in these settings requires visibility into and control over (a) lower layers; (b) entities that are not "just" networks, including CDNs, service/server/CDN brokers, and storage systems; and (c) endpoints of the end-to-end communication, possibly including NICs, video players, and content servers.

These commonalities suggest a research agenda that looks at how to jointly optimize across different layers and different entities, which may have control loops that function at different granularities and with different goals. This goal will require an understanding of the APIs and abstractions that enable the optimization. In a cross-domain (and perhaps commercial) setting, research will also be necessary to understand how to incorporate business models and brokering that incorporates these models. To provide a platform for this research, the community would benefit from infrastructure with rich monitoring that exposes these layers, and from testbeds capable of realistically exercising the various components of end-to-end delivery.

Transformative research areas

1. Cross-layer optimization

Large portions of our computing and network infrastructure are becoming software-defined -- our infrastructure is no longer limited by rigid, closed hardware, but key portions are being programmable via flexible software interfaces, and easily virtualizable and slice-able. In particular, we can now flexibly control and configure aspects of end-to-end flows at multiple different layers, including both low level and high level issues such as: what kind of and how much compute resources they use (e.g., in the cloud or on mobile system), the specific backend systems flows invoke (e.g., personalization engines), the network paths they traverse and the specific attributes of those paths (e.g., the latency and bandwidth, or waypoints traversed, the physical layer attributes), what kind of transport to use (e.g., leveraging in-network support vs. not), whether or not hardware support is leveraged to accelerate packet transmissions, and how the content being accessed is actually delivered (e.g., transcoded vs. not, encrypted vs. not, from a local cache vs. a remote data center). These attributes can be changed on the fly as needed according to, e.g., a user or application's need, much better than today. Such flexibility can form the basis for ensuring optimal end-to-end quality of experience.

To realize this vision however, a key issue is providing opportunities for cross layer visibility and optimization. In the absence of such scheme, we may end up with silo-ed solutions which suffer from many issues: (1) techniques developed for one layer or resource cannot be generalized or applied to other layers, (2) because of lack of cross-layer visibility, the techniques can result in locally optimal, but globally sub-optimal decisions and (3) the techniques can have undesirable interactions counter-acting their individual benefits.

We need research both into *mechanisms* -- what APIs to offer that enable cross-layer coordination across storage, networking, compute and the application -- and *policies* -- what algorithms result in ensuring optimal coordination, at scale.

2. Integrating different types of entities, each with control loops operating at different levels/goals

The control planes used by sophisticated applications, such as video delivery and cloud computing, to manage computation, storage and communication currently operate in a completely independent fashion from those that manage network connectivity . However, the decisions to perform processing tasks on different machines, store content on different nodes or allocate bandwidth to different flows all have significant impact on each other. As we move forward, we expect the desire for better performance and reliability to drive tighter integration between these control planes – both in the form of coordination between them and control planes that manage multiple of these resources in a more integrated fashion.

The problem of integrating multiple control planes is that we lack the appropriate interfaces to coordinate their decision making process. Ideally, increasing the visibility of each control plane into the decision process of its peers would help greatly. However, in practice, this is difficult for a number of reasons. First, competitive concerns limit the amount of information any participant is willing to release. Second, it is likely unreasonable for every participant to understand the metrics and tradeoffs of its peers. This is especially true in designing interfaces between control planes that operate at different levels of the architecture and have different objectives. For example, it has proven difficult to define standard metrics between routing domains and it is likely to be even more difficult to define metrics that CDNs and ISPs can agree to.

This does not mean that defining cross-domain control plane interfaces are hopeless. Often, a small number of end-to-end application tie these control planes together and recent efforts in understanding user QoE provide a target optimization goal for the end-to-end communication path. However, we need significant work in exploring the design of these interfaces to help understand the tradeoffs involved in optimizing end-to-end user QoE while managing the concerns of the parties involved. It may prove valuable to begin exploring these issues in a few well-defined settings such as video delivery and cloud computing to help identify the key properties needed in any cross-domain setting.

3. APIs and abstractions

Software-defined infrastructure implies a profound transition: from a largely application-agnostic network, whose behavior is determined by a mix of operator policy and vendor decisions, to a network whose behavior is primarily under the control of the application designer. This leads to an immediate question: what APIs and abstractions should be presented to the application designer?

A helpful analogy is to consider storage systems. The layout of data on disk is vital to a storage system designer; this has inspired the rise of read- and write-oriented filesystems, data warehouses, column-oriented databases, and NoSQL and object-oriented databases, among many other things. But control of layout is indirect. Few if any storage system designers directly lay out blocks of each file or object on disk. Rather, an architecture is chosen which determines the overall layout of blocks for an arbitrary object on disk, and the schema of each object is defined in terms of this architecture. The designer does not know precisely how the blocks of an object are laid out on disk, but the architecture gives a close enough approximation that he can estimate the storage performance (write time, read time, query time) quite well.

This is the level of abstraction that a network application designer needs: sufficient control over and visibility into the network's handling of his application flows that he can get tight estimates on end-to-end performance, but not detailed information on the exact routing of each packet.

It is important to note that these abstractions go well beyond routing and bandwidth control at layer 2 and layer 3. Far more important is *computation siting*. We are moving into a world where it will be possible to site computation *anywhere*, and this is by far the most impactful network optimization that can be imagined; all of the routing control and bandwidth optimization in the world doesn't come close to the performance gain offered by moving the program next to the end-user or the data.

So this is a reasonable first cut: an abstracted network with bandwidth and latency specified between users, data, and points of computation, and the ability to site computation anywhere within the network (or at least anywhere within the network where there is an open computational POP). This first cut is only that, of course: the real question is how one does this at scale, in the presence of partitions, both long-lived and transient, and at network

speeds. The implementation of this abstract API also implies some infrastructure obligations: not simply OpenFlow Networking rules, but an orchestration abstraction that instantiates and activates programs throughout the network. In addition to the standard Cloud tools (Fabric, Ansible, Puppet, Chef) one should also look at extensions to the scientific workflow engines for this (Kepler, Pegasus, XSEDE).

4. Incorporating business models as part of the interaction

To study interaction across players/providers, a difficult question is how to model the pricing/cost model of providing a service, i.e. what is the cost of providing/brokering a service of a certain quality and how it affects prices and competition among players, how to capture incentives to cooperate to provide an end-to-end service. Here it would be interesting to create models, then try them out in "living labs" with real people.

Infrastructure requirements

1. Operational infrastructure that exposes rich monitoring

We need a federated testbed with compute resources spanning data centers, end-hosts and mobile devices, software-defined networking and storage support at different locations, and software-defined exchanges. In addition, we would need the ability to conduct deep instrumentation of the experimental setup to drive various optimizations.

The testbed infrastructure should support large-scale experimentation on end-to-end delivery of network services and applications. For applications and services developers, the testbed should provide simple APIs to access the capabilities (compute, storage, bandwidth resources) of the underlying infrastructure. In particular, efforts must be made to enable experimentation on the testbed by researchers from other scientific and engineering disciplines. For researchers working on developing and improving the testbed infrastructure itself, rich monitoring capabilities should be provided by enabling direct access to the state information of the infrastructure. Such capabilities should include information on current network status, link utilization, compute/storage node usage, controller response times, physical layer/channel properties and debugging/error messages. Trained support staff must be available to help users with adopting the infrastructure for both research and education.

2. Realistic conditions to engage users and services

The testbed(s) should support all expected technology areas that students and households will use in 5-10 years (compute, storage, networking, mobile, ...), with deep instrumentation to collect data. Then we can engage real users (e.g. students or residents of SDI communities) in novel services. This will provide a degree of realism in the service experiments, and open the door to many forms of QoE experiments.

3. Measurement and control across all layers

For the testbed infrastructure to truly enable innovative research focusing on ideas beyond the Internet today, it is critical that the testbed support experimentation at all network layers including the physical layer (e.g. optical fiber communications and networks) interconnecting the distributed facilities. For example, increasing levels of traffic and QoS/QoE requirements of next-generation applications on the Internet dictate the need to investigate bypass mechanisms which avoid overhead due to IP packet processing. Such seamless migration of traffic to lower layers in the network stack must be possible in the testbed infrastructure. Integrated multi-layer control frameworks should be developed to operate the testbed infrastructure efficiently.

Measuring and Monitoring

Participants: Rudra Dutta (chair, North Carolina State University), Xenofontas Dimitropoulos, Dan Kilper (University of Arizona), Balachander Krishnamurthy, Cees de Laat (UvA), Yan Luo (Univ. of Massachusetts Lowell), Ben Mack-Crane (Corsa Technology), Nagi Rao (ORNL), David Stern (DISA), Mike Zink (University of Massachusetts Amherst).

Measurement faces several challenges at present: (1) Scaling challenge: it is non-trivial to provide detailed per-flow measurement data at a line rate of 100+Gbps, and collecting the metrics on every core in a data center is an enormous task; (2) Diversity and dynamics of measurement tasks: the measurement target could change rapidly, so do the interested metrics; and (3) Knowledge discovery from measurement data: huge pools of measurement data are underexploited assets that could help learn interesting patterns, predict trends and lead to actions.

These challenges have the additional dimension that additional usefulness may exist for collected measurement data that only becomes apparent at a future time. An example of such a challenge is the use of measurements to provide cross-domain accountability. Measurements originally collected for the purpose of routing performance monitoring and fault detection, say in a multi-provider-hop data transfer context, may later be mined to study how well the various providers held up the user's expectation regarding emergent characteristics such as jitter, Mean Opinion Score of video quality, etc. With the post-facto application of such analytics, it may also become apparent that a specific measurement, currently not being collected, may add crucial value to the existing measurement streams in light of a new analytics intent. The measurement system for an SDI needs to be at least as agile and as programmable as the SDI itself.

We need software-defined measurement - programmable monitoring and measuring (M&M). Perhaps the right place for it is in a software-defined measurement control plane. How do you map a single M&M task that is coming from the control plane by data plane devices with completely different capabilities? How do we mine measurement data? Across devices? And based on this, how can we close the loop by apply measurement data, e.g., perform machine learning for continuous application and infrastructure optimization and control, and as needed, programming new measurement tasks.

The new SDI world gives rise to an abundance of deep research questions regarding measurement, such as:

- Future infrastructure will be managed and controlled by machines, so the information that it needs to control and operate must be machine interpretable. What are the right information models, and how do they include the meaning and units of what one is measuring?
- Will deep learning, machine learning and big data methods will be used to gather and process that information?
- It is key to research what granularities are needed in measurement. At some point the measurement and metering can become a huge overhead. What is needed for operation and exception/fault finding, and what is needed for "sending the bill"?
- Information modelling and programming: Program language objects that represent metering & monitoring that allow programs to be written to extract the desired information.How can we understand and ensure the stability of controlled systems?
- How can we monitor and measure applications themselves, so that you know the effectiveness of the delivered SDI slice?

Transformative research areas

1. How does one understand what and how to measure?

SDN/SDI requires newer network-level and also higher-level quantities such as dynamics of controllers, impact on end physical systems, etc.; some are measured directly and others need to be extracted using analytics. We have some current understanding of network-level quantities: latency, bandwidth, jitter, and these can be

accommodated in a perfSonar-like infrastructure. However, even now we have many important measurements that today must be inferred, e.g., the response time of controller and devices, scalability with respect to number of flows, how fast service is provisioned/restored. As we move to more complex systems, especially those which are cyber-physical systems (CPS), we are encountering whole new "measurement surfaces." In these new systems, how do we know what to measure? Are models and analytics (such as used in formal experiment design) useful tools? Can machine learning help humans to better understand what to measure?

2. What are the right dynamics for measurements?

Future measurement tasks will be highly dynamic due to the flexibility of SDI and heterogeneity of resources and applications. The creation, change, and determination of measurements depend highly on the objectives and infrastructure capabilities. Hence the measurements should be programmable and dynamically configurable at different levels.

The metrics, targets, timing, frequency and so on are dimensions to define a measurement task, and the question exists on who the authority is to determine a measurement task. Research is needed on how to dynamically define, program and map measurement tasks onto heterogeneous programmable instruments that exist in a broad context of cyber-physical systems. The boundary between the hardware and software support for measurement is dynamic and changing as technology advances, calling for research efforts.

3. Cross-domain measurement and incentives for information-sharing

As services span multiple administrative domains, there is a need for measurements that verify that a given domain is delivering what it has promised. However, service providers usually prefer to reveal as little information as possible about their infrastructures. How can these two desires be reconciled? Are there systems of incentives that will make this easier? Incentives need not only be financial. Users may choose to share monitoring information (that they otherwise consider private) simply in order to have a voice in the design evolution of a system, if they are assured and convinced that their feedback will inform the design evolution. Can users instantiate measurement probes along their slices to help verify that they are received the promised infrastructure support? Since information-sharing will be automated like measurement itself, it must have commonly understood semantics. It is possible that the system itself can provide capabilities to allow domain authorities or other entities to pre-program policies regarding what incentives the entity is willing to offer, allowing an automated agent to even tailor the incentive for specific information exchanges. All such exchanges should be preceded by informed consent, but it is a difficult question as to when a human authority is truly informed (requiring a user to read through a dozen pages of a clickwrap agreement is likely not an effective way of informing them). These are complex socio-technical questions that will need to engage economists and sociologists in addition to computer scientists.

Further, such cross-domain measurement and information sharing immediately raises the semantic problem of comparing apples to apples. One provider's definition of a quantity as basic as "throughput" may not match another's. This can become crucial when using measurements to produce accountability. When an optical signal crosses over from one provider's fiber to another, to whom can any resulting impairment be assigned? If reduction is noticed in a higher layer's performance, can it be ascribed to an impairment introduced at a lower layer, or is it a shortcoming of the higher-layer resident on the lower layer? Such questions already go unanswered in today's traditional fiber and wireless networks, and as new technologies such as Free-Space Optics, 60 GHz, Thz etc. come into being, and interplay in a large SDI, will become crucial unless specifically addressed. Today's measurement systems utilizing agents at measurement points and measurement collectors stop short of assigning commonly accepted semantics, assuming that the producer of the measurement information and the consumer share a common understanding, while others do not need to understand these measurements. This assumption is no longer true in a cross-domain M&M scenario, and commonly shared semantics are critically needed for future SDI.

Infrastructure requirements

1. Shared methodologies and repositories for measurement data and tools

A data archival solution (similar to what CRAWDAD is currently offering for wireless community) should be made available to the SDX/SDI community. This would allow easy sharing of the data and benefit the larger research community. In addition to an archive that simply stores measurement data the community should think about what additional meta data should be saved with such data to enable others to reproduce measurements and experiments.

2. SDI testbeds must be extensively instrumented, with all results open and public

All future testbeds that are build to support SDI/SDX research as well as existing testbeds that might be extended should incorporate instrumentation. In addition to the instrumentation tools should be created that will allow researchers to conduct experiments and collect measurement data. With instrumentation and appropriate tools SDI/SDX testbeds should be widely usable by the research community. Ideally, the proposed tools can be connected with the above mentioned archive such that data from an experiment can be directly archived. Likewise, this connection to the data archive should also allow experimenters to use archived data for their experiments. E.g., to generate network traffic based on data that was measured in a different experiment.

SDXes and the Internet Model

Participants: Tom Lehman (chair), Marco Canini, David Du, Arpit Gupta, Ethan Katz-Bassett, Bryan Larish, Joe Mambretti, Ibrahim Matta, Bruce Patterson, Dave Reese, Srinivasan Seshan, Joe Touch, and John Wroclawski.

This session focused on the relationship between the Internet model and SDXes. This topic was explored from the perspective of SDXes adding value within the general framework of the current Internet architecture. This topic was also explored from the perspective of how SDXes, and related technologies, may lead to an entirely different global communications infrastructure that is radically different in form and function from today's Internet. In an attempt to identify transformative research areas associated with both of these perspectives, at set of questions were identified to guide the discussions of the specific architecture and technology areas.

- How do SDXes fit into the current Internet architecture and model? What value would they add in this context?
- What would a future SDX and SDN/SDI dominated global communications architecture look like? What functions would it provide? How would it provide all that is available from today's Internet? What extra features would it provide that would be sufficient to motivate a transition?
- What are some specific SDX services and functions that are needed to realize both of these visions?

The general consensus in the room was that SDXes, in combination with SDN/SDI, may well be a transformative technology. It was also generally agreed that fundamental research and development should be conducted from both of the perspectives noted above. For the 5 year time horizon, research into how SDX technologies can be applied to components of the current Internet architecture is needed to establish the value of these concepts. In parallel, longer term focused research into a more transformative vision for a global communications infrastructure based on SDX and SDN/SDI should be pursued.

Four main transformative research areas were identified in pursuit of these goals.

- **SDX Transformation of Current Exchange Points:** This research area is concerned with how to transition the relatively static Exchange Point configuration and functions to a system that is truly software defined and programmable. For the commercial Internet the Internet Exchange Points (IXPs), this is largely associated with BGP configurations and policy. For the Research and Education (R&E) community Exchange Points are more focused on lower layer network peering establishments. As these exchange points evolve into a software defined infrastructure, the distinction between the commercial and R&E exchange points may diminish. For this research area, it is largely assumed that IP connectivity will remain as the primary mechanism for globally scalable any-to-any communications. The SDX features will enable new services that exist in parallel and add value to the current Internet architecture.

- **New Exchange Point Architectures:** This research area is focused on looking beyond the current exchange point functions and transforming the exchange point architecture and services. An SDX is envisioned which has core functions of deep programmability, sliceable, hierarchical, multi-layer, and recursive properties. This type of SDX architecture could be the basis for providing value added within the context of the current Internet architecture, as well as a transition to a new global communications infrastructure.

- **Distributed SDX Orchestrated Control:** This research area is focused on how to realize end-to-end value added functions via orchestration and control of a distributed SDX ecosystem. A single SDX may provide value to the specific location where it resides. However, truly transformative functions will require the development of higher-level end-to-end services that leverage the distributed programmable SDX

features in coordinated fashion.

- **SDX Based Global Communications Architecture:** This research area is focused on how SDXes may be a key component of a new global communications infrastructure. The research is motivated to define an SDX/SDI based architecture that is globally scalable for any-to-any communications, and can also provide rich sets of new programmable services.

Two requirements for infrastructure to support these research and development efforts were also identified. Additional details on these research areas and infrastructure requirements are provided below.

Transformative research areas

1. Investigating how SDXes will transform today's Internet exchange points

The Internet ecosystem has changed dramatically in past decades. With growing demand for video content, large content providers like Akamai, Google, etc. are now peering with as many networks as possible and at as many locations as possible. Even though the Internet structure is evolving with growing demands, the underlying routing protocol BGP is not. In this new ecosystem, network operators desire more flexible and programmable control over their traffic. Such flexibility is not possible with BGP and researchers have been exploring how to overcome the limitations of BGP---enabling flexible and programmable traffic control. Given that the network operators around the globe have invested heavily in border routers that use BGP to advertise routing paths, we have to explore an incrementally deployable solution. A solution which can provide the desired programmability and flexibility with minimal deployment cost, maximal impact, and ensures interoperability with other BGP routers. We developed iSDX: an industrial scale SDX platform for IXPs with these goals in mind. iSDX uses SDN as a tool to allow fine-grained programmable traffic control at IXPs.

BGP was originally designed to advertise reachability information and it is not suited to perform traffic engineering tasks. In future, we envision Internet architecture to be a composition of SDN based forwarding policies, expressed at multiple SDXs and default BGP forwarding. SDX can make sure that BGP routers are used for the basic task of advertising reachability, and SDXs can be used to provide the flexible and programmable traffic control.

The idea of replacing the traditional layer-2 switch at IXP with a programmable SDN switch sounds like a very simple idea. Though making sure that such an SDX platform scales to the production traffic at large IXPs is non-trivial. There are three main challenges involved:

- Programming Abstraction: How can we ensure that IXP participants program their control plane policies independently?
- BGP Interoperability: How can we make sure that the flexibility at SDX is not coming at the cost of creating forwarding loops and black holes on the Internet?
- Scalability: How can we ensure that SDX can combine SDN policies of hundreds of participants with the default forwarding policies in a feasible manner?

These issues were addressed in "SDX: A Software Defined Internet Exchange" (presented at SIGCOMM 2014) and "iSDX: An Industrial-Scale Software Defined Internet Exchange Point" (to be presented at NSDI 2016). We used various techniques from the distributed systems to scale the control plane computations and developed new data plane encoding techniques to scale the data plane. Building a production ready SDX platform allows us to explore various applications that can be built on top of this platform. So far, we have developed few SDX applications: application-specific peering, inbound TE, server load-balancing etc. From our interactions with multiple IXP operators and participating networks, network security emerged as the most exciting SDX application. Examples include:

- Preventing BGP Hijacking:
  - SDX can disable propagation of unauthorized BGP routing advertisements---dampening the impacts of such attacks.

- Detecting and mitigating DDoS Attacks:
  - SDX can prevent propagation of spoofed (source) traffic by enabling ingress filtering (BCP 38) at the IXP.
  - SDX can also monitor traffic at the IXP to detect DDoS attacks with wider footprint. Such detections can triggers flow table entries to mitigate such attacks at much finer level of granularity.

2. Creating new kinds of exchange point architectures

The Internet is based on a layered endpoint model and a network model based on global addresses and local forwarding decisions. This model has been evolving to become increasingly virtual and hierarchically composed, e.g., including tunnels as links (overlays and VPNs) and logical routers (e.g., LISP), as well as hierarchies specific to routing protocols (BGP) and management (ASes). SDXs need to support a similar level of hierarchy and encapsulation to both match these Internet endpoint and network models, as well as to support scalability. Viewing this hierarchical composition more generally as a recursive capability not only supports the existing Internet architecture, but emerging recursive architectures that enable dynamic layer composition, protocol hierarchy composition, policy management composition, and service/function composition, which can result in more agile and adaptive capabilities than the current Internet supports.

Future SDX infrastructure needs to support a model where end system processes and protocol stacks can be managed orthogonally to network infrastructure. End system processes need to be able to independently coordinate with multiple SDI instances, where each instance attaches at a OS communication terminus - e.g., a "socket" in Unix-style systems. End system protocol stacks need to be able to independently coordinate with multiple SDI instances to enable fine-grained control of end system protocol optimization as well as to support cross-process and cross-protocol/protocol-instance coordination, e.g., for striping/channel-bonding or ensemble/temporal context amortization. In a sense, SDIs need to mature to become peers in the the independent mutual coordination between end system processes and end system protocols that are already part of modern OSes.

Additionally, SDXs need to seamlessly interoperate with end system services to coordinate the deployment of host services (e.g., processes) and independently couple them to network resource attachment points (e.g., network interfaces). This independent coupling provides distinct advantages over today's "slice" model, as it enables interlayer translation and internetwork gatewaying that are already widely used aspects of the existing Internet model. It also allows more fine-grained and orthogonal configuration of both communication and processing components of a network system.

Supporting independent peer coordination between end system processes, end system protocol stacks, and SDI requires SDX extensions to support distributed, peer-to-peer agreement and coordination mechanisms. These include support for asynchronous interaction, accommodating varying views of message order, nested transaction locking, and other aspects of more byzantine agreement and optimization.

3. Orchestrated control of distributed SDXes

SDX is an emerging concept and we are in the very early stages of discussion and debate about the optimal architecture, functions, and services. However, it does seem clear at this point that coordination amongst a distributed ecosystem of SDXes is where the high value killer applications are likely to be found. Several possibilities were noted in the area of cyber attack detection and defeat. Other possibilities were discussed around using SDXes as a vehicle to develop an innovation ecosystem where application owners could tailor what happens to their data flows and system topologies. This may be thought of as "democratizing the middlebox functions and infrastructure". A research area defined as "Distributed SDXex Orchestrated Control" was identified as an area that should be pursued as a vehicle to determine how to utilize a distributed ecosystem of SDXes to realize new advanced cyberinfrastructure services. The following key research topics are noted as components of this broader topic:

- **SDX Resource Descriptions, Advertisements, and Discovery.** From the SDX resource owner's perspective, technologies and standards are needed to enable abstracted resource description and service provisioning in the context of voluntary and ad hoc federation participation. The described

resources need to include network/compute/storage and the services available. These resource descriptions will likely require sophisticated realtime modeling technologies to allow the resource owners to control how, when, and at what level their resource descriptions are made available to external agents. This will require research and development into methods for realtime automated construction of resource description models that have features to allow resource owner tailoring of the abstraction level and also integrate local policies.

- **Distributed SDX Service Definition.** Services need to be defined in the context of end-to-end multi-domain SDX functions that support users and user agents.
- **Distributed SDX Dynamic Service Level Federation.** Service level federation systems need to be developed that provide the proper security, authentication/authorization granularity, and dynamism. The idea here is that federation needs to be realized at the service level, as opposed to the infrastructure level. As compared to current federation technologies, these federations need to be an order of magnitude more dynamic, automated, and tailorable to specific sets of users and services in a realtime context.
- **Distributed SDX Orchestration.** Multi-domain orchestration technologies/protocols are needed to work thru the issues of service provisioning in a distributed SDX environment and autonomous operator environments. This may require addressing issues of realtime information hiding and just-in-time sharing, scalability, security, multiple autonomous system resource negotiation, and state rollback.

4. Investigating issues arising in the vision of an SDX-based global architecture

Can an SDN based architecture evolve to become globally scalable? If so, how? What issues will arise? We currently have a working example of a global communications / computer architecture, namely the Internet. What lessons can we learn from the development and use of today's Internet?

Infrastructure requirements

1. Testbed that interconnects SDI/SDX prototypes with legacy networks at multiple points

Provide access to a distributed SDX testbed which supports experimentation as a independent AS and has peerings at real commercial IXPs and R&E Exchange points.

Not all pieces of the Internet will evolve at the same rate or in the same way, and so emerging services/protocols will have to interact with legacy networks and protocols, including BGP. Therefore, we need to experiment and prototype in a mixed setting with realistic settings. We have limited visibility into the Internet's topologies/policies/traffic, and hence models and isolated testbeds have limited fidelity in capturing the Internet.

To supplement these models and mitigate their limitations, the community would benefit from a testbed that interconnects with the real Internet. This testbed infrastructure should allow researchers to emulate an autonomous system(s) that can exchange routes and traffic with the actual Internet, subjecting experiments to real Internet topologies, policies, and traffic. Given the rich interconnectivity of today's widely peered Internet (e.g., some networks peer with thousands of other networks) and the richer options that may result from adoption of SDX technologies, such a testbed must connect with many networks at many locations, including IXPs around the world, and it must be flexible enough to support a range of experiments and services. The testbed should be sliceable and support multiple experiments and researchers in parallel. It should support mechanisms to allow software/SDX control of experiments even at legacy Internet locations. The infrastructure should also permit federation with other testbeds.

2. Sliceable, programmable, modular SDX infrastructure which includes integrated network, compute, storage resources.

Provide access to a distributed SDX infrastructure where the SDX technology is based on deep programmability, sliceable, hierarchical, multi-layer, and recursive properties.

A key objective is to develop a design for a highly distributed programmable, sharable environment, within which component resources can be discovered, assembled, integrated, utilized, and then discarded after use. Within this environment, component functions and capabilities must be appropriately abstracted, within a hierarchy of capabilities. Consequently, the core infrastructure foundation must be comprised of programmable, configurable resources accessible through high level well known standard APIs, including APIs that are integrated within orchestration processes. All of the basic building blocks of the core infrastructure must be selected because they conform to the objectives of the environments, e.g., programmability, abstraction, sliceability, etc.

The SDX infrastructure needs a more modular interface to end system capabilities, including both OS processes and the end system network stack. Currently popular “slice” models bind SDI instances to individual processes in ways that undermine cross-instance interaction. These current models support communication that ends at the process, but this overlooks the need for SDX to coordinate more closely with the protocol stack in the end system.

Securing SDI, and SDI for Security

Participants: Jim Griffioen (Chair), Alvaro Cardenas, James Chacko, Russ Clark, Nick Feamster, Rodrigo Fonseca, Yang Guo, Hongxin Hu, Hyojoon Kim, Cees de Laat, John Moore, Mehrdad Moradi, Glenn Ricart, Munindar Singh, Byrav Ramamurthy, Malathi Veeraraghavan, and Tilman Wolf.

This session began with lightning talks given by Alvar Cardenas, Hongxin Hu, Mehrdad Moradi, and Yang Guo. Session attendees were given time to write down their thoughts, followed by a time of discussion where each person explained their comments. The session ended with a lively discussion regarding the three transformative research challenges needed to secure SDI infrastructure.

The discussion began by exploring the security issues that must be addressed in the design of an SDI-based network infrastructure. As the session progressed, the discussion began to turn toward the topic if utilizing SDI network infrastructure to offer improved security to users and their applications. As a result, the group took the liberty to change the title of this session (see above title) to include the use of SDI for enhancing security. Tossed in throughout the discussion were concerns and challenges related to establishing trust and shared/cooperative management of the SDI infrastructure (in a secure way). These discussions lead to the grouping of transformative research challenges into three broad areas:

- Securing the infrastructure
- Using new technologies in SDI to achieve security guarantees for applications
- Establishing trust and cooperative security arrangements

Transformative Research Challenges

1. "Being Secure" - Securing SDI/SDX hardware and software infrastructure

While a software defined infrastructure is highly flexible, adaptable, and can evolve and change quickly, it raises a long list of new security challenges that must be addressed to ensure that the software control structures and APIs to the infrastructure are not susceptible to attack. Although it is not immediately obvious that an SDI physical infrastructure creates any new challenges that were are not present in existing physical network infrastructures, this is an area that needs to be explored to ensure there are no new vulnerabilities. Securing the SDI control system (e.g., controller software/server) is a key area of research, including devising identity and authentication/authorization mechanisms for interacting and gaining access to the software control infrastructure. The protocols used to interact with the infrastructure must be secure. The ability to access measurement data has been brought up in several other sessions as an important feature of SDI networks. However allowing access to the measurement system raises new security concerns about who can access the measurement data, which data they can access, and possible attacks on the measurement infrastructure itself as it becomes an increasing important part of the SDI infrastructure (e.g., needed for the feedback and analysis loop outlined in the keynote).

Slicing was identified as an excellent way to isolate and protect users, and limits the the attack surface because many potential attackers can't even send their own attack packets into the slice. Security attention is then focused on admission control for the slice which is a smaller and more auditable set of code. However, research is needed in this area to assure that slices are set up securely with appropriate admission control, that slice identification is unforgeable (and impersonation is impossible), that lower layers including hardware properly enforce slice boundaries, and that heterogeneous systems and SDXes can securely match slice identification across the SDX and/or heterogenous system interfaces. We also need the ability to specify appropriate levels of security per slice (e.g., all inter-component data must be encrypted with specified protocols and keys), that policy distribution and key distribution can be securely accomplished as part of slice creation, and that we have appropriate verification models and auditing mechanisms. All of these things will require research programs, prototypes, and guidelines/standards. In addition to information protection and privacy, we also need to consider the performance impact of slices on each other. Performance isolation of slices is yet another area for research.

Identifying the correct languages for expressing security policies is another key area of research as well as the software and APIs used to map policies to rules and eventually insert policy into the infrastructure. Research into automatic policy analysis and verification is also needed.

Research into the overheads associated with making the system secure. There is a tradeoff between security and performance that must be considered and explored. This also involves understanding the potential attack vectors and developing models to explore the performance/security tradeoff.

2. "Delivering Security" - Using SDI to enhance the security of network applications

Although an SDI infrastructure raises new security challenges, it also has the potential to enable new security services inside the network that benefit applications. Identifying the security abstractions and services that the SDI infrastructure should provide to applications will be key to creating an infrastructure with enhanced security features that far exceed what we have today. One can imagine a variety of security services and abstractions offered by the SDI infrastructure, including offering Security as a Service (SaaS). The slice abstraction itself provides isolation that could be leveraged to create more secure network applications. Fully understanding the power and limitations of the slice abstraction as it relates to security is an important issue. Current SDI infrastructure largely employs tuples (e.g., source/destination addresses and ports) to ensure an application's flows are treated a particular way (e.g., securely, or with some performance guarantee). However using tuples is an indirect way to ensure security for applications. Having a more direct way to identify and protect an application's flows is needed (e.g., a cryptographic token associated with the application carried in packets). Clearly additional research is needed to identify abstractions that enhance security for applications.

Moreover, security functionality in the current internet is largely achieved via the use of middleboxes (e.g., firewalls, IDS/IDP, proxies, etc). In an SDI network these functions can be virtualized (e.g., via network function virtualization (NFV)) and dynamically created and placed where needed. This raises a variety of challenging questions regarding elasticity, placement, effectiveness and correctness, etc. Cooperation between virtualized, dynamically instantiated, security functions also becomes a problem in such systems.

3. "Shared/Cooperative Security" - Social, legal, economic, regulatory policy, and trust challenges

Security is not merely a technical problem. We must state from whom we are securing the infrastructure, and for what purpose. In other words, there isn't going to be a blanket solution for all stakeholders and application scenarios. How can we enhance the design space and explore that enhanced space effectively to find custom solutions for each setting? Security considerations are inherently challenges of dealing with autonomy. When we take a systems view, it often becomes clear that security violations arise primarily because of a misalignment between the social and the technical architectures. It is crucial to expand the design space to incorporate the social elements. A major research challenge is in finding ways to explore this design space with respect to competing objectives, especially, "liveness" (users being able to accomplish the services that interest them) and "safety" (prevention of bad states). Traditionally, we tend to focus on safety, but it often leads to users either not achieving what they want (a denial of service, even if not caused by an attacker) or taking ad hoc measures to circumvent technical security measures, which is usually worse (because it is ad hoc) than having knowingly relaxed some requirements.

SDI provides an opportunity to customize the solutions for each application and the concomitant set of stakeholders. Emerging governance approaches, especially those that incorporate economic reasoning, provide a way to expand and explore the design space so we can offer the level of security that is appropriate for each application scenario and stakeholders involved in it. These approaches presume an ability to encode subtle requirements relevant to security (not exclusively traditional security requirements in that they would include other requirements, both functional and nonfunctional.

End-to-end security the most important issue is how to create distributed trust across domains in a scalable way. If one is breached or under attack the help of others may be needed. Information may need to be exchanged, and policies may need to be updated and exchanged to avert the attack. The trust must be based on something. That might be [money, law, incentives, peer to peer, etc.]. In a world of cyberphysical systems, where the amount of domains, players, may reach millions - billions, scalable and machine interpretable ways are necessary. Such

systems may also show new behaviors including autoimmune reactions, etc. So there are quite a number of socio/technical research questions that must be addressed.

A significant challenge in securing SDI is the fact that many parts of the infrastructure itself may be shared among multiple participants. For example, at an SDX, many participating independently operated networks may have access to a shared switch fabric. Each of these participating networks may write policies concerning how they want traffic at the exchange point to be handled. This mode of interaction presents several immediate concerns:

- What permissions and authorization should each participant have to read and write shared infrastructure?
- What is the mechanism for expressing access control policies?
- Who or what should be responsible for enforcing these authorization and access control policies?

One example might involve control over traffic flows. Consider the "traffic scrubbing" services that many commercial service providers offer. These mechanisms often involve re-routing traffic that is destined for a particular network through some intermediate network—an action that the intermediate network might trigger in response to alerts concerning the presence of a denial of service attack. The SDI infrastructure needs a way of determining that such re-routing of traffic is allowed at all, that re-routing to that particular intermediary is permitted, and that the particular request to re-route the traffic was made by an authorized party.

Another challenge in securing SDI concerns how conflicting policies can and should be resolved. Given a shared SDI infrastructure, individual participants (e.g., autonomous systems) may express conflicting policies concerning how traffic should be forwarded. For example, a network that is sending traffic may specify a particular outbound traffic policy, whereas the recipient of traffic may specify a conflicting inbound traffic policy. The SDI must incorporate mechanisms for detecting and resolving conflicts in ways that respect the authorization and privileges of individual participants.

Infrastructure requirements

1. Translation of physical security policies to virtual infrastructure

What changes need to occur to current physical security procedures (e.g., physical access logs, physical configuration changes, etc), in light of the fact that the key components that need to be secured are now software infrastructure (which could be running anywhere).

The infrastructure must include security at multiple levels. Traditional security architectures begin with physical security (e.g. building access, room, cage, rack) all the way up through the hardware and software systems.

Part of this architecture is based on best practices for achieving security but it is also largely motivated by audit and reporting requirements. All of this is much more difficult in a "software defined" world where things are virtualized and highly distributed. This becomes even more difficult when this virtual infrastructure spans multiple management domains. It is important that we design and develop audit and reporting capabilities into our security architectures in the SDI models.

2. Both an SDI research infrastructure and a security experimental infrastructure are needed

At least two types of SDI infrastructure will be required to support exploration of security issues related to SDI networks and services. For lack of a better name, a safe and secure "research production" infrastructure is required by most researchers to experiment with new SDI services and applications. The existing GENI network is an example of such an infrastructure. While it is used for research and is not guaranteed to be reliable, the system offer some level of basic service and protection from outside attacks. A second "security experimentation" infrastructure (similar to DETER) is also required to explore and investigate the types of security issues described above. Testing and evaluating security functions in a virtualized environment requires an isolated cloud platform

It is critical to secure the testbed infrastructure before actually deploying it for real-world services and applications. The testbed should itself not become an enabler for launching cyberattacks either internally or

externally. An infrastructure that enables multiple security services providers to operate on top of basic network infrastructure to promote competition would be useful.

Virtualizing SDI in the Datacenter and Cloud

Participants: Theo Benson (chair), Andy Bavier, Raj Jain, Raj Kettimuthu, Aditya Akella, Anduo Wang, Ben Yoo, Haoyu Song, Ilya Baldin, Inder Monga, Jerry Sobieski, KC Wang, Madeleine Glick, Mark Berman, Orran Krieger, Rick McGeer, and Zhi-Li Zhang.

This breakout session discussed the roles and hierarchy and recursion in large-scale, virtualized environments; how we can rethink the factorization of infrastructure and application functions as we move beyond today's Network Function Virtualization trends; the important role of administrative domains and how SDI applications will likely span multiple such domains; and the growing need to include widely heterogeneous devices (e.g. sensors) in tomorrow's version of the cloud.

Transformative research areas

1. Multi-scale clouds (Hierarchy of clouds)

With mobile edge computing, carriers are putting microclouds on the tower. With VANETs, each car will have a microcloud. Also, most homes have (or will have) a microclouds. So clouds will be everywhere and applications will span multiple clouds. This is also called fog computing. The clouds here are not only storage devices but also provide computation for time-critical applications. The most time-critical tasks will get done at the edge and more compute-intensive less time-critical tasks will be moved to bigger clouds at higher tiers.

Thus, multiple clouds of multiple capability under multiple domains will be used by applications. This raises an interesting set of research issues starting with scheduling that takes care of latency which is highly dynamic and needs certain guarantees. Interfacing with multiple resources using their own APIs. The resources include computation, and storage in addition to networking. There issues of heterogeneity, security also arise.

2. Rethinking and refactoring the functions of applications / infrastructure

Virtualizing various network functions will be an essential and integral part of a software-defined infrastructure. Today most of these network functions are implemented in expensive, specialized hardware network appliances or "middleboxes" that are deployed in various part of the network, compute and storage infrastructure. Many of these network functions are application, service or provider-specific, with complex control logics, opaque specifications and vendor-specific interfaces. Network function virtualization does not simply entail "porting" network functions that are currently implemented in hardware middleboxes into software modules running on commodity servers in a data center or a cloud environment. It offers opportunities to rethink and refactor network functions as an integral part of a software-defined infrastructure instead of as add-on "software middleboxes" bolted on top of a software-defined infrastructure. Therefore network function virtualization poses important research challenges both in terms of developing new abstractions for seamless integration of network functions into a software-defined infrastructure and designing flexible SDI control architectures that effectively incorporate network function control that can be optimized for higher-level metrics such as application/service performance.

While the flow-based, "match-action" abstraction is adequate for a conventional network data plane (e.g., with openflow switches) with packet forwarding as its main functionality, such an abstraction is not sufficient for a far richer data plane of a software-defined infrastructure that incorporates complex network functions. The operations on data packets required by diverse network functions will go beyond simply looking up and matching certain header fields, perhaps rewriting some of them or inserting a few new header fields. For example, some network functions, e.g., DPI (deep packet inspection) function for malware detection, encryption/decryption functions or transcoding function, require examining or transforming the payload of data packets, and touch every packet of a flow. Many network functions are stateful in that control decision logic depends not only on information contained in a single packet but across multiple packets within a flow, or information exogenous to the flow. In addition, the notion of a flow may no longer make sense for certain network functions, the operations of which may alter end-points of flows dynamically or involve multiple flows belonging to a single application or service. New abstractions that go beyond today's flow-based, "match-action" data plane abstraction are clearly

needed to support flexible network functions. Furthermore, new control plane abstractions for incorporating application, service or provider-specific control logics and stateful decision making are also called for.

Refactoring network function architecture is also crucial: for instance, what network functions should be incorporated into the "core layers" of a software-defined infrastructure, what should be delegated or relegated into the provider-specific "infrastructure fabric" layers, and what should be left at "application" or "service" layers while at the same time maintaining some level of control and transparency into their control policies. What are design principles, theories and algorithms for deciding functional decomposition and placement? Finally, what design and implementation principles are required to turn abstractions into practical and scalable frameworks, e.g.,in terms of open reference implementation (ORI), to demonstrate their feasibility and correctness?

3. Multi-domain infrastructure

Due to difference in administrative domains, there is limited visibility across domains: How do you deal with debugging? How do you assign blame? How do you make assertions?

The prospect of fault localization in a multi-cloud multi-domain environment, where visibility into and control of various participating intermediate domains are limited, poses substantial challenges to future research. The discussion centered around at least two key propositions: First, the notion that it is desired to expose as much of the lower layer control parameters as possible for inspection by external domain agents so that detailed analysis can take place. And second, the specification of service definitions (SLAs) that could define the service expectations being delivered by each such domain.

The notion of "domain" was itself recognized as having different overloaded definitions: Administrative (policy) domain, service domains, technology domains, and even legal or jurisdictional domain over which or through which the application services are delivered. How do we understand services / applications that span multiple legal domains, e.g., differing national privacy laws? How can such important issues be incorporated into our notions of applications and services?

4. How to support heterogeneous physical infrastructure in a cloud

What are the right kinds of explicit models/interfaces for exposing resources? How do you define attributes of resources within the model? What are the right primitives for expressing the resources? What is the right level to expose resources? How do the primitives allow users to create other higher level resources? What sort of algorithms do we need to safely virtualize the physical infrastructure? How do we ensure the algorithm terminates?

Currently, storage is poorly modeled: how do we accurately model storage? Specifically how do we ensure isolation? How do we capture the effects of mediation or contention?

In 5-10 years, we expect to see massive "data torrents" pour into clouds, providing an abundance of real-time data. These torrents originate from devices that can be programmed and tasked (e.g. weather radars). How do we represent and use such devices as part of a cloud service?

Infrastructure requirements

1. Continued development of open-cloud environments

Current cloud commercial cloud offerings are based largely on virtualized interfaces to compute, storage, and networking. However, these interfaces are usually opaque, and provide no visibility into how they map to underlying resources. Such visibility can be important for tenants to achieve several high level goals such as performance, bounded latency, independent failure, or security. Fortunately the NSF Cloud program has begun to provide experimental infrastructure that provide "bare metal" provisioning for experimental cloud systems, and enables research on these and other issues that can benefit from co-optimization of the virtual layer. These systems are a good start, but need to be larger, more distributed (e.g. with edge clouds), and contain a wider variety of devices (such as high-bandwidth sensors) to explore real-time data management. Their instrumentation and toolsets also need to be expanded and improved.

2. Infrastructure for collecting, sharing, and using torrents of data

One of the defining challenges of upcoming cyberinfrastructure will be learning to cope with large amounts of different types of data. These data torrents may arrive from individual high-volume sources (e.g., high bandwidth scientific instruments), from the aggregation of large numbers of sources that produce relatively modest individual data streams (e.g., millions of environmental sensors), or from combinations of sources along this spectrum. The data that comprises these torrents will come with varying handling requirements implied by ownership, privacy concerns, value, and reliability. Research into mechanisms for data storage, dissemination, analysis, anonymization, and multi-party computation relies on experience with and availability of a variety of such data torrents drawn from real data sources. The working group recommends the placement of small "cloudlets" at strategic locations to serve as sources for data torrents. These cloudlets, possibly modeled on a GENI rack with augmented storage, should include sufficient resources not only to collect the data, but also to provide torrent storage and sufficient computational resources to execute *in situ* data analyses, reduction, anonymization, etc. Such torrent cloudlets should have reachback connectivity to larger data center or cloud resources to permit more computationally intensive processing as needed.

The Cloudlets need not be at fixed locations. For example, one participant noted that an automobile will soon have more computational horsepower than an InstaGENI rack, at least in terms of number of cores. This cloudlet is of course mobile, and has sporadic connectivity to boot; further, it has a wealth of high-bandwidth sensors, and in-situ reduction is the most reasonable way to deal with all that data. A reasonable view of the future is that each car is a Cloudlet; and one interesting question is what applications will be written for this class of Cloudlet. One key, as this example illustrates, is that the Cloudlets must be open. A decade ago a phone was just, well, a phone. Today it is a portable supercomputer, running applications that were not imagined a decade ago. The number of apps in the Apple app store has gone from nothing to 1.5 million in a decade. Today a car is, well, a car. In a decade it will be a Cloud that moves. How many Cloud apps will be in the Tesla App Store?

There are many interesting candidates for data torrents. The working identified several but believed that these are only representative and other sources should be pursued. Specific types of data torrent discussed include sensor data from automobiles or automobile fleets, medical sensor data ranging from radiology equipment to wearable sensors, agricultural sensor data, satellite imagery, and municipal or state/provincial transportation data, and data from smart devices which collect it as a side effect of their operational purpose: for example, the data from smart parking meters can yield a wealth of data for transportation and commerce in a city.

Opt-in data torrents are also of interest. Every resident of every advanced nation now carries in his pocket a supercomputer with 4G and sporadic WiFi connectivity, and a dozen or so sensors. Simple applications can harvest the information from these sensors to provide pictures of movement, energy use, communication patterns, etc. While there are obvious privacy concerns with harvesting this information, opt-in donation of information does not excite these concerns. This is particularly the case if it is coupled with anonymization.

3. Storage resources

We need to better understand the right primitives and abstractions for supporting data transfers and compute placement in federated clouds: how do you make reservations? how do you adapt reservations to utilization? How do you understand and manage storage and file systems in heterogeneous, multi-domain environments?

Storage performance isolation requires partitioning/virtualization along several dimensions: IOPs, storage CPU processing capacity, network link connecting storage to the attached node and the bus/memory channels connecting the storage network interface to CPU/Memory. This is not always achievable, especially with off-the-shelf storage systems, especially when it comes to isolating the network traffic of one storage volume from the others (e.g. being able to associate a storage volume with a network channel, like a VLAN or a well-defined flow).

Parallel file systems that stripe data (blocks that belong to the same file or object) across multiple disks add more challenges. In order to provide guaranteed performance for a single file/object access, multiple physical disk resources need to be provisioned and each of those physical resources are shared by several files/objects. When constructing testbed-oriented storage implementations it is important to combine high performance with as many isolation properties as possible to support proper performance and security isolation of storage elements of the

infrastructure from one another. Particularly challenging in this aspect may be parallel storage implementations, like Ceph or Gluster - the testbed storage system should be flexible enough to accommodate experimentation with existing and new parallel storage implementations while supporting improved isolation.

To support such requirements, infrastructure needs to provide storage guarantees/isolation. We need APIs and UIs (and hopefully GUIs) that present flexible access to the storage. This should be backed by infrastructure that supports the different level of flexibility.

Appendices

Appendix A: Participants

- Aditya Akella *(UW-Madison)*
- Alvaro Cardenas *(University of Texas at Dallas)*
- Anduo Wang *(UIUC)*
- Andy Bavier *(Princeton University, ON.LAB)*
- Arpit Gupta *(Princeton University)*
- Bala Krishnamurthy *(AT&T Labs - Research)*
- Ben Mack-Crane *(Corsa Technology)*
- Ben Yoo *(University of California, Davis)*
- Bruce Patterson *(City of Ammon, ID)*
- Bryan Larish *(Georgia Tech)*
- Byrav Ramamurthy *(University of Nebraska-Lincoln)*
- Cees de Laat *(University of Amsterdam)*
- Chip Elliott *(Raytheon BBN Technologies)*
- Dan Kilper *(University of Arizona)*
- Darleen Fisher *(NSF)*
- Dave Farber
- David Du *(University of Minnesota)*
- David Reese *(CENIC/Pacific Wave)*
- David Stern
- Ethan Katz-Bassett *(University of Southern California)*
- Fontas Dimitropoulos *(FORTH (Crete), Greece)*
- Glenn Ricart *(US Ignite)*
- Grant Miller *(NITRD)*
- Haoyu Song *(Futurewei Technologies)*
- Hongxin Hu (*Clemson University)*
- Hyojoon Kim *(Princeton University)*
- Ibrahim Matta *(Boston University)*
- Ilya Baldin *(RENCI/UNC Chapel Hill)*
- Inder Monga (ESnet)
- Jack Brassil (NSF)
- James Chacko *(Drexel University)*
- James Griffioen *(University of Kentucky)*
- Jerry Sobieski *(NORDUnet)*
- Joe Mambretti *(ICAIR Northwestern University)*
- Joe Touch *(USC/ISI)*
- John Moore *(Internet2)*
- John Wroclawski *(USC ISI)*
- KC Wang *(Clemson University)*
- Ken Calvert *(University of Kentucky) - unable to attend*
- Larry Landweber *(Raytheon BBN Technologies)*
- Madeleine Glick *(University of Arizona)*
- Malathi Veeraraghavan *(University of Virginia)*
- Marco Canini *(UC Lovain, Belgium)*
- Mario Gerla *(UCLA CSD)*
- Mark Berman *(Raytheon BBN Technologies)*
- Mehrdad Moradi *(University of Michigan)*
- Mike Zink *(University of Massachusetts Amherst)*
- Munindar Singh *(NCSU)*
- Nagi Rao *(Oak Ridge National Laboratory)*
- Nick Feamster *(Princeton University)*
- Orran Krieger *(BU)*
- Padma Krishnaswamy *(FCC)*

- Raj Jain *(Washington University in Saint Louis)*
- Raj Kettimuthu *(Argonne National Laboratory)*
- Rick McGeer *(CDG/US Ignite)*
- Rob Ricci *(University of Utah)*
- Rodrigo Fonseca *(Brown University)*
- Rudra Dutta *(NCSU)*
- Russ Clark *(Georgia Tech)*
- Srini Seshan *(CMU)*
- Theo Benson *(Duke University)*
- Tilman Wolf *(University of Massachusetts Amherst)*
- Tim Talty *(General Motors)*
- Tom Lehman *(UMD/MAX)*
- Yan Luo *(University of Massachusetts Lowell)*
- Yang Guo *(NIST)*
- Zhi-Li Zhang *(University of Minnesota)*

On the following pages:

Appendix B: Call for Participation

Appendix C: Workshop schedule

Call for Participation

Software Defined Infrastructure / Software Defined Exchange Workshop

Organized by the NSF "Looking Beyond the Internet" Planning Group
Workshop chairs: Rob Ricci, University of Utah
& Nick Feamster, Princeton, University
ricci@cs.utah.edu; feamster@cs.princeton.edu

Abstract

The NSF "Looking Beyond the Internet" planning group led by Chip Elliott, Dave Farber and Larry Landweber is conducting a series of workshops intended to discuss new research opportunities in the broad areas of future wireless, networks and clouds. The goal is to engage the research community to identify potentially transformative network architectures, enabling technologies and applications in three broad categories: software-defined infrastructure, community-scale wireless networks and future applications and services.

The purpose of this workshop is to recommend a 5-year agenda for research programs in new technology paradigms that will radically transform the Internet, and to identify the needs for infrastructure to support that research.

We are now at the dawn of Software Defined Infrastructure, in which systems that used to be implemented by rigid control systems or in hardware are now increasingly programmable and virtualizable: the result is that the systems become much more open to transformative research with implications for revolutionary new applications and services. Today's early examples include multi-tenant clouds, software defined networking, network functions virtualization, and software defined radios. Individually, these systems present large research challenges, and these problems are compounded when they are interconnected into end-to-end, Internet-scale systems.

Looking forward, the emerging concept of software-defined exchanges will enable large-scale interconnection of Software Defined infrastructures, owned and operated by many different organizations, to provide logically isolated "on demand" global scale infrastructure on an end-to-end basis, with enhanced flexibility and security for new applications.

This workshop will identify research problems that must be solved before this vision can be realized. Concrete recommendations will be made for associated research and infrastructure support programs. Participants will be active researchers and infrastructure developers.

Areas of interest for this workshop include, but are not limited to:

- Software-defined infrastructure and software defined exchanges, broadly defined
- Prototypes of software defined exchanges

- Interoperability issues for software defined exchanges
- Applications and services enabled by software defined infrastructure and exchanges in a multi-domain environment
- Federation of multi-domain software defined infrastructure
- Resource slicing in multi-domain software defined infrastructure
- Operating systems for software designed infrastructure
- Security issues for software defined infrastructure and exchanges
- Interconnection of software-defined networks and NFV with other types of resources
- Inter-cloud and other interconnections that cross domains

As a pre-requisite for workshop attendance, contributions are solicited from prospective participants in the form of one-page white papers identifying new research directions and related testbed infrastructure requirements. The whitepaper should also briefly describe your background in this area. The white paper should be submitted at `sdi-workshop.flux.utah.edu` by Nov 20, 2015. Those accepted for the Workshop will be notified by Dec. 1, 2015

VENUE / DATE

Washington DC, Date: Feb.4-5, 2016

FUNDING

Limited funds are available for travel and local expenses of attendees. Guidelines will be sent prior to the Workshop. Please indicate in your submitted white paper if you wish to be reimbursed.

SDI/SDX Workshop Agenda

Thursday, February 4				
8:00	–	8:30	Breakfast	*Provided*
8:30	–	8:45	Welcome: Workshop goals and logistics	Workshop chairs
8:45	–	9:15	Keynote I	Nick Feamster
9:15	–	9:30	Breakout instructions	Workshop chairs
9:30	–	12:00	Breakout Session I	*Four in parallel*
12:00	–	1:00	Lunch	*Provided*
1:00	–	1:30	Keynote II	Marco Canini
1:30	–	4:00	Breakout Session II	*Four in parallel*
4:00	–	5:00	Outbriefs from Breakouts I and II	Breakout chairs

Friday, February 5				
8:00	–	8:30	Breakfast	*Provided*
8:30	–	9:00	Keynote III	Orran Krieger
9:00	–	11:30	Breakout Session III	*Four in parallel*
11:30	–	12:00	Outbriefs from Breakout III	Breakout chairs
12:00	–	1:00	Conclusions from the workshop	Workshop chairs
1:00			Boxed lunches	*Provided*

Breakout rooms

Washington Floor 4, suite 400. *The same meeting room used for plenary sessions*

Dupont Floor 7, suite 750. *To the right of the elevators. Upon entering the suite, the first conference room to your right.*

Mt. Vernon Floor 9, suite 900. *Inside the Internet2 suite.*

Farragut Floor 9, suite 900. *Inside the Internet2 suite.*

Breakout format

Breakout topics, chairs, and attendees will be determined in advance of the workshop. Participants will vote online on interesting topics for breakouts, and the workshop chairs will pick breakout chairs and assign attendees to breakouts.

Each breakout starts with 4-5 "lightning" talks of 5 minutes each, no slides. Each attendee who submitted a whitepaper will give a talk in one of the breakouts. The remaining 2 hours are for discussion and synthesis.

The primary outputs of each breakout are to be:

1. A set of N (suggested: $N = 3$) areas of research likely to be interesting 5–10 years out:

- A ‘bullet point’ statement of the area
- The identity of someone assigned to write a short piece about it

2. A set of M requirements for infrastructure, data, other ‘non-research’ prerequisites for doing the work

Note: The “topic” assigned for the each breakout is a starting point, not a hard and fast rule. It is perfectly acceptable for the areas of research to stray from that starting point. However, the workshop is focusing on research in the years 2020 and beyond, and that focus should be maintained.

Outbreif format

Each breakout chair gets 5 minutes to present the outputs of the breakout. The remainder of the time (~10 minutes) is used for the audience to suggest common themes that are seen in more than one of the breakout outputs, and to identify the most pressing issues.

Final conclusion session

The goal of this session is twofold:

1. Pick a set of research areas and infrastructure requirements to highlight in the final report. (All will be included, but we want to highlight a ‘top N’ set.)
2. Pick a set of themes (identified in the outbriefs) that seem the strongest and/or most common to highlight in the final report.

Breakout Session I (Thursday Morning)

Models for programming and verifying SDI

Room: **Mt. Vernon** Chair: **Aditya Akella**

- **Anduo Wang:** *Software-Defined Networks as Databases*
- **KC Wang:** *How to Derive Abstraction for Software Defined Infrastructure and Software Defined Exchange*
- **Rick McGeer:** *Declarative Verifiable SDN Specification*
- **Theo Benson:** *Commoditizing the "S" in SDN with BareMetal Switching Infrastructure*
- Aditya Akella
- Dan Kilper
- James Chacko
- Joe Touch
- Mehrdad Moradi
- Srinivasan Seshan

SDXes in practice (experience with SDX and SCI-DMZ deployments)

Room: **Dupont** Chair: **Yang Guo**

- **David Reese:** *Pacific Wave SDX*
- **Hyojoon Kim:** *ScienceDMZ and IPS Bypass*
- **Jerry Sobieski:** *Thoughts on Research Topics and Tools for Future Research*
- **Joe Mambretti:** *Next Generation Infrastructure Based on SDI, SDN, and SDX Architecture and Technologies*
- **Russ Clark:** *Software Defined Infrastructure in the AtlanticWave SDX*
- **Tom Lehman:** *Software Defined Exchange (SDX) and Software Defined ScienceDMZ (SDSDMZ) Ecosystem*
- Arpit Gupta
- Balachander Krishnamurthy
- Bruce Patterson
- David Stern
- Haoyu Song
- Yang Guo

Co-evolving applications and infrastructure

Room: **Farragut** Chair: **Mike Zink**

- **Bryan Larish:** *White Paper Response to Call for Participation: SDI/SDX Workshop*
- **Glenn Ricart:** *Whitepaper from Glenn Ricart*
- **Nagi Rao:** *Co-Designed SDIs for Computing and Physical Complexes Using Integrated Analytics*
- **Rudra Dutta (NCSU):** *Human-Network Interaction: the Weakest Link?*

- **Zhi-Li Zhang:** *Integrating Application-Aware Virtualized Network Functions in Software-Defined Infrastructure*
- Andy Bavier
- Ben Mack-Crane
- David Du
- Ibrahim Matta
- Madeleine Glick
- Mario Gerla
- Mike Zink
- Orran Krieger
- Raj Kettimuthu
- Rodrigo Fonseca
- Yan Luo

Connecting across domains (policies, resource brokering, etc.)

Room: **Washington** Chair: **Tilman Wolf**

- **Ilya Baldin:** *SDI Research Challenges and Opportunities*
- **John Moore:** *SDN Infrastructure Ecosystem*
- **Ken Calvert:** *Enhancing Network Services Through SDXs*
- **Malathi Veeraraghavan:** *Multi-domain, multi-layer software-defined infrastructure*
- **Mark Berman:** *Transforming Scientific Collaboration via Software-Defined Infrastructure*
- **S. J. Ben Yoo:** *A Market-Driven Broker Plane for Multi-Domain Software-Defined Infrastructures*
- Alvaro Cardenas
- Byrav Ramamurthy
- Cees de Laat
- Ethan Katz-Bassett
- Hongxin Hu
- Inder Monga
- James Griffioen
- John Wroclawski
- Marco Canini
- Munindar Singh
- Raj Jain
- Tilman Wolf
- Tim Talty
- Xenofontas Dimitropoulos

Breakout Session II (Thursday Afternoon)

The future of programmable network hardware

Room: **Mt. Vernon** Chair: **Marco Canini**

- **Ben Mack-Crane:** *Understanding and Managing Change in an SDN*
- **Dan Kilper:** *Deep programmability for optical systems in the future Internet*
- **Haoyu Song:** *Open Programmable Data Path: Towards White Box 2.0*
- Balachander Krishnamurthy
- Bryan Larish
- David Stern
- Hyojoon Kim
- Jerry Sobieski
- Marco Canini
- Yang Guo

SDI as a marketplace

Room: **Dupont** Chair: **Rodrigo Fonseca**

- **Bruce Patterson:** *White Paper Response to Call for Participation: SDI/SDX Workshop*
- **Cees de Laat:** *The Service Provider Group Framework*
- **James Griffioen:** *SDXs as Resource Marketplaces*
- **Munindar Singh:** *Software-Defined Governance: Applying Computational Norms*
- **Tilman Wolf:** *Application for Participation in Software Defined Infrastructure / Software Defined Exchange Workshop*
- Glenn Ricart
- John Moore
- John Wroclawski
- Ken Calvert
- Orran Krieger
- Rodrigo Fonseca
- S. J. Ben Yoo

Intereconnecting different technologies: mobile, optical, and storage

Room: **Farragut** Chair: **Mark Berman**

- **David Du:** *Beyond the Internet: Convergence of Networking and Storage Becomes A Must*
- **James Chacko:** *Software-Defined Communication Testbed*
- **Madeleine Glick:** *Shaping and quality of service in SDN controlled hybrid optical/ electrical networks enabled by machine learning*
- **Mario Gerla:** *A wireless SDN inter exchange for mobile systems*
- **Tim Talty:** *The Role of Mobile Assets in Future Software Defined Infrastructure*
- Alvaro Cardenas

- Anduo Wang
- Hongxin Hu
- Joe Touch
- KC Wang
- Mark Berman
- Mehrdad Moradi
- Rudra Dutta (NCSU)
- Russ Clark

End-to-end: SDI across the wide area

Room: **Washington** Chair: **Ethan Katz-Bassett**

- **Aditya Akella:** *Toward end-to-end software defined QoE*
- **Byrav Ramamurthy:** *Balancing Exchange Points Communication and Inter-controller Communication for Inter-Domain Routing*
- **Inder Monga:** *End-to-end, multi-domain SDN with SDI bookends*
- **Srinivasan Seshan:** *Cooperation Between Control Planes*
- **Xenofontas Dimitropoulos:** *Stitching Inter-Domain Paths over IXPs*
- Andy Bavier
- Arpit Gupta
- David Reese
- Ethan Katz-Bassett
- Ibrahim Matta
- Ilya Baldin
- Joe Mambretti
- Malathi Veeraraghavan
- Mike Zink
- Nagi Rao
- Raj Jain
- Raj Kettimuthu
- Rick McGeer
- Theo Benson
- Tom Lehman
- Yan Luo
- Zhi-Li Zhang

Breakout Session III (Friday Morning)

Measuring and Monitoring

Room: **Mt. Vernon** Chair: **Rudra Dutta**

- **Balachander Krishnamurthy:** *What we can learn from OpenFlaw*
- **Mike Zink:** *White Paper for Software Defined Infrastructure / Software Defined Exchange Workshop*
- **Yan Luo:** *Challenges and Opportunities of Measurement for Future Software Defined Infrastructure*
- Ben Mack-Crane
- Cees de Laat
- Dan Kilper
- David Stern
- Nagi Rao
- Rudra Dutta (NCSU)
- Xenofontas Dimitropoulos

SDXes and the Internet model (layering, the end-to-end argument, AS model, etc.)

Room: **Dupont** Chair: **Tom Lehman**

- **Arpit Gupta:** *An Industrial-Scale Software Defined Internet Exchange Point*
- **Ethan Katz-Bassett:** *Real Internet Experiments for Future Internet Services*
- **Ibrahim Matta:** *BU submission*
- **Joe Touch:** *The role of state and layering in software-defined networking*
- **John Wroclawski:** *The Tiger Meets the Termite Mound: Toward Robust Inter-Domain SDN*
- Bruce Patterson
- Bryan Larish
- David Du
- Ken Calvert
- Marco Canini
- Srinivasan Seshan
- Tom Lehman

Securing SDI

Room: **Farragut** Chair: **Jim Griffioen**

- **Alvaro Cardenas:** *Secure Software-Defined Federated Networks*
- **Hongxin Hu:** *Virtualizing and Utilizing Network Security Functions for Securing Software Defined Infrastructure*
- **Mehrdad Moradi:** *Exploring Security and Mobile Operator Use Cases for SDX*

- **Yang Guo:** *SDI Workshop*
- Byrav Ramamurthy
- David Reese
- Glenn Ricart
- Hyojoon Kim
- James Chacko
- James Griffioen
- Joe Mambretti
- John Moore
- Malathi Veeraraghavan
- Mario Gerla
- Munindar Singh
- Russ Clark
- Tilman Wolf
- Tim Talty

Virtualizing SDI in the datacenter and cloud

Room: **Washington** Chair: **Theo Benson**

- **Andy Bavier:** *CORD: Central Office Re-architected as a Datacenter*
- **Raj Jain:** *Software Defined Cloud Virtualization for Multi-Cloud Applications*
- **Raj Kettimuthu:** *Software Defined Infrastructure: Challenges and Opportunities*
- **Rodrigo Fonseca:** *Towards a Network Marketplace in a Cloud*
- Aditya Akella
- Anduo Wang
- Haoyu Song
- Ilya Baldin
- Inder Monga
- Jerry Sobieski
- KC Wang
- Madeleine Glick
- Mark Berman
- Orran Krieger
- Rick McGeer
- S. J. Ben Yoo
- Theo Benson
- Zhi-Li Zhang