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

Chairs:

Robert Ricci (University of Utah)

Nick Feamster (Princeton University)

February 4–5, 2016

Washington, D.C.

Summary

IPurpose of the Workshop

We are now at the dawn of Software Defined Infrastructure (SDI), where 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.

Software-defined infrastructure and software-defined exchanges (SDXes) have the potential to affect the way that we design not only those systems, but the fundamental architecture of the Internet itself. The purpose of this workshop was 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. It also looked at infrastructure projects that could help to catalyze this research.

This workshop was part of an NSF series of "Beyond the Internet" workshops and was held on February 4–5, 2016 in Washington, D.C.

Likely Areas for Transformative Research

Through white papers, talks, breakout sessions, and open discussion, the workshop identified a number of areas where the participants believe that transformative research may be found in the five-to-ten-year time frame. A full catalog of these areas can be found in the "Breakout Sessions" portion of this report. Here, we highlight a set of N themes that are common across these areas, and which suggest areas of high impact in the field of networking and beyond.

- **Abstractions for Delivering and Reasoning about SDI**

  - Specifically across provider and user domains (limited information)
  - OS-like abstraction of services, coordination of resources
  - Abstractions to reason about
    - Performance

        - Security
        - Individual components (including heterogeneous technologies)
        - Composition of individual components and modules
        - Layering
        - Network-wide (and cross-domain) Services
        - Federations of domains
    - In support of easy programming, verification and formal modeling, capturing a broad range of devices and services, economic interaction, and highly dynamic interactions
    - Policies and security properties that are understandable by humans and the network [PL and HCI]
    - Developing (and programming) programmable data planes

- **The SDI Control Architecture**

    - Taking advantage of emerging technology trends
        - Programmable mechanisms to perform measurements on and extract state from the network (e.g., in-band network telemetry). Richer data sources in general.
        - Scalable, distributed statistical inference and automation [analytics, ML, maybe HCI problems] to inform...
        - Customizable actions such as protocol-independent network elements to drive...
        - Fully programmable data planes, which provide the ability to customize actions [connect to infrastructure]
    - Modularity
    - Architecture of data flow
    - Multi-party/Multi-domain control
    - Designing and taking advantage of next-generation data-planes

- **Relationships between Networks (broadly defined) and Users / Applications**

    - To be written by who
    - Mutual optimization problems [needs model of what we are operating and objective functions] [needs incentives to measure / share]
    - SDX-like infrastructure moving out to the edge
    - Security largely about trust and actively supporting user privacy
    - Economic and regulatory concerns

- *Other ideas: where do these fit?*
    - *ISA for the SDI dataplane*
    - *AI assisted management*
    - *Safe upgrading of the data plane*

- *Attack surface*

Building an Environment for Transformative Research

The workshop also looked at what national-scale infrastructure projects might help to provide the impetus and infrastructure for this research; this is infrastructure that could be built in the short to medium term in order to catalyze long-term, forward-thinking research ...

- **Platform to help research get "out in front" of commercial capabilities**
  - To be written by who
  - *What route?*
    - *More performant software forwarding?*
    - *Specialized, but not fully custom, hardware platform?*
    - *FPGA-based?*
    - *Fab?*
  - Should it include hardware?
- **Experimental Testbeds**
  - To be written by who
  - Hardware, software, or both?
  - Software: reusable shared software base
    - Sharing of drop-in components
    - Create compelling, unique, maintained software artifacts
  - For specific purposes, distributed
  - *Fast*
  - *Real use (Markets = money) [eat own dogfood]*
  - *Widely deployed*
  - *Diverse*
  - *Beyond networks (computation, storage, sensors, actuators, 5G, cars)*
  - *Multiple domains [connectivity to external domains]*
  - *Instrumented*
  - *What we have, do we need something new*

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.

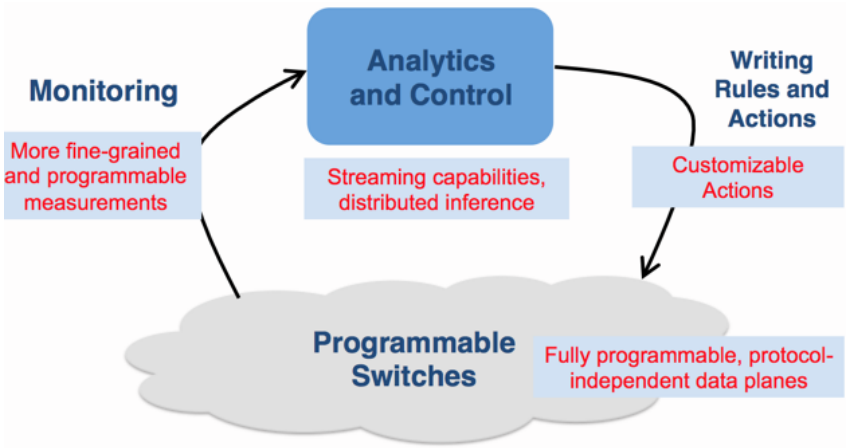
Where We've Been. The retrospective portion of the talk motivated the need for SDXes in interconnected networks where different portions of the network are operated by different (and independent) administrative domains. Although the public Internet is a prominent example of such a network, the talk emphasized 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.

Breakout Sessions

Models for Programming and Verifying SDI

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.

Written by *Aditya Akella*

Transformative research areas:

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

The vision of SDI and SDX portrays a paradigm where a collection of diverse types of network connected entities can all be programmatically created and controlled to achieve different, often application-driven purposes. In addition to the network itself, compute, storage, data, sensors, cyber physical systems (e.g., vehicles), etc. can all be viewed as part of the SDI. 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.

Written by *KC Wang*

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?)
Written by *Rick McGeer*

3. Instruction Set Architecture for the SDI dataplane

SDN currently supports a small, fixed set of operations. 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. This begs typical concerns over the instruction set:

- Is there benefit to grouping sequences or sets of basic SDN configurations 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 SDN 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.

Written by *Joe Touch*

4. Delegation: slightly smarter physical elements (switches) vs. hierarchical control

Should we define what should go where? Is a better approach to declaratively specify high level objective(s) and let a program synthesis-driven technique automatically generate code?

In SDN, there are some network 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 SDN networks if we push some of these tasks to data plane switches. In general:

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

Text to be written by *Theo Benson, Mehrdad Moradi*

Infrastructure requirements:

1. Underlying hardware

Highly programmable devices. Support efforts that are aimed at building such specialized hardware. Vendors may not want to build these. E.g. NetFPGA

- can't experiment with control-plane/data-plane functionality boundary tradeoffs without the ability to put more functionality in the data plane
- current switch hardware is limited in programmability
- software switches support greater flexibility but are too slow for production settings
- 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.
  Text to be written by *Srini Seshan, Dan Kilper*

2. Deployment

Deployments consisting of ranges of switches - white box to highly programmable? How and where are these deployed - racks? Across wide-area? Near an IXP? Near testbeds with significant compute?

From a verification perspective, it is critical that the actual deployment allow researchers to

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.
- Provide significant hardware diversity to stress the synthesis task. This requirement is motivated by the ongoing challenges faced during the shift of SDI switches from OpenFlow API to the more expressive P4 interface. 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.
- Support a pool of relatively homogeneous hardware virtual slices so that synthesis tools can explore the space of platform-specific optimizations.
- 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.

Text to be written by *Theo Benson, Srini Seshan*

SDXes in Practice

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

Research and Education (R&E) cyberinfrastructure is in the early stages of a major transformation being driven by the emergence of Software Defined Infrastructure (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.

A vigorous community debate is underway regarding what exactly are the defining features and capabilities of this new SDI based cyberinfrastructure. 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 ScienceDMZ 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.

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. 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.

2. A more stable software stack, from the controllers to storage and software infrastructure.

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

4. Protocol-independent switch ASICs, to support rapid development of more flexible packet processing pipelines in switches.

Education requirements:

1. Education and training of next-generation of network engineers, who must be trained not only to think about network protocols, but also software development.

The following paragraphs do not belonging to the breakout discussion of SDXes in Practice. Will be removed if no one claims them.

Notes/Input:

To ultimately achieve photonic/sdn interface where engineers do not have to know details of technology on other side of the interface, education and research to bring requirements of applications and networks to photonics community and capabilities of photonics to network community, which also implies educating networking community concerning photonic capabilities (apart from simply replacing copper wires with optical fibers) [this also may apply to wireless]. This should be attached to specific research goals to encourage ongoing interaction.

Following paragraphs written by *Chip Elliott,* ***suggest they go somewhere in this report*** *(this section?)*

As we move from today's Software Defined Networking to the emerging vision of Software Defined Infrastructure, 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. Furthermore, 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.

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.

Major research challenges arise with this vision. 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?

Co-evolving Applications and Infrastructure

A summary of the session, to be written by *Mike Zink* (zink@ecs.umass.edu)

Attendees:

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

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. Interconnecting software-defined infrastructures

Text will discuss: Matches, preserves, emulates, or transforms layer 2 slices across the interchange

- type (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

Highly reliable SDX architectures

Text to be written by *Glenn and others*

2. Human-Manageable, AI-assisted SDN/SDI

Text written by *Rudra*

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) [Alin2005, Nanog23], Google users' loss of service [Ars Technica, 2012/12], the infamous June 2013 Ztomy DNS misconfiguration [Schultz2013, Cisco blog].

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 network and cyber-infrastructure systems 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 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

Text to be written by *Nagi Rao*

SDI technologies must be developed to encompass end systems ranging from small physical systems to large science instruments, from tablets to supercomputers and clusters. Complex data flows may be setup on demand or in-advance to encompass a variety of components.

- Mario's thoughts could easily go in here as well (comment by Rudra)

4. Performance specification, quantification and standardization

Text to be written by *Nagi Rao*

There has been very rapid development of SDN controllers, devices and applications. It is unrealistic and unmanageable to choose the right combination of components for a solution. Performance figures of each of these components must be provided with them. One approach is to assess the impulse response of the components within a standard test harness.

- Rudra: this section could possibly be integrated with the next one. Look upon the thing as performance/monitoring information scheme design; as part of that, look at what parts can be safely exposed to what other partners/players/competitors, and with what incentive.

5. Design of Cooperative Network-App information sharing scheme

Topic suggested by *Rudra*

… based on discussion mainly from Rodrigo, Orran, Rudra, Larry, Raj. What are the right abstractions that will let various levels of virtualized networks (SDI substrate provider, application-specific SDI, application designer, application user) exchange information that will help performance all around, but not require exposing more information than any is willing to (for reasons of business models, privacy concerns, whatever)?

- Declarative interfaces was suggested by Rodrigo, so as to not expose unnecessary details among layers and enable intra-layer optimization

- Economic models suggested by Zhi-Li and Rudra; as a possible model for co-designing and co-evolving application/infrastructure by keeping in mind incentives

… functionality and data abstractions must be complemented with performance such as throughput, responsiveness, scalability and security (in particular, protection against hanging oneself unintentionally with controllers)

Ibrahim phrased it as the need to think in layers and define high level interfaces for applications to express requirements, which then get compiled/optimized/placed into the infrastructure.

- Madeleine spoke about the need for people who understood both sides of a layer boundary, in ORDER to be able to design the right abstractions that hide the layer details from each other (with the example of optical comm devices that could be well utilized by SDN)

Larry commented that a great research challenge could be: "what are the abstractions for the cyberinfrastructure of 5 /10 / 20 years out?" What are the abstracted interfaces we will be speaking (as today we speak of the abstractions "packets, layers, end-to-end" etc.? What are the things we will need to standardize, so that we can interoperate? Slices, SDIs, SDXs? These will have to include things like trust, manageability?

Orran - SDN has enormous promise of optimizing the application by controlling the network. Fundamentally today it has just been a tool to empower the system administrator at a particular layer. Even if the underlay today 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.

Infrastructure requirements:

1. Need for Testbeds with Real Applications and Real Users: Metro SDXes

Text to be written by *Glenn and others (feel free to write/edit here!)*

Comment by Orran could go in here or be part of this? "Must include real user populations, and they have to have buy-in"

2. Test harness to assess and quantify performance (controller, device, app)

Text to be written by *Nagi Rao*

Standardized tests may be designed and implemented under test harness in a testbed. New technologies can be simply inserted into testbed and their performance figures can be attached to them. 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.

3. **Testbed that incorporates research level photonics which would also serve as a basis and focal point for education and bringing communities together** (suggested by Madeleine)

4. **Testbed Infrastructure for Education**

5. Continuing to support the community and bringing it together

Connecting Across Domains

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

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 Software Defined Infrastructure architecture (federated, multi-domain, etc.) and complexity management (modularity, composability, coordination functions); inter-substrate management

To date, most Software Defined Infrastructure 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:

A. 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?
B. 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?
C. 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.

D. 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?
E. 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 model 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, heterogeneous, shared, and sliceable testbed infrastructure is needed to support robust experimentation with new concepts in scientific workflows and collaborations. (vendor relationships, global scale); Heterogeneous testbed: vehicles, optical, IoT; include storage and computation

The infrastructure that supports this research agenda needs to be viewed as a system that encompasses campuses, regional networks, 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. Infrastructure features, reusable software

Testbeds need to be built that will allow researchers to experience, experiment 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 re-usable software components that deal with common testbed features. GENI is one example of such a project. As people build newer SDN 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 GENI like project.

The Future of Programmable Network Hardware

This session was attended by Ben Mack-Crane, Dan Kilper, Haoyu Song, Balachander Krishnamurthy, Bryan Larish, David Stern, Hyojoon Kim, Jerry Sobieski, Marco Canini, Yang Guo, Yan Luo. We 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 two 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. We also identified two sets of main infrastructure needs to catalyze research along these themes.

A summary of the session, to be written by *Marco Canini*

Transformative research areas:

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

Text to be written by *Dan Kilper*

Smart cities and other large scale enterprises will rely on a wide range of networked devices and associated heterogeneous network technologies including software defined radio, visible light communications, fiber optic networks and wavelength switching, packet, circuit and flow switches. Greater integration at the device, sub-system, 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.

[Haoyu Song] Create chips that can be easily sliced and can be interactively programmed per slice.

2. Safety and verification of programmable devices, including how to safely upgrade the data plane

Written by *Ben Mack-Crane*

(Prompt: Safety and verification of programmable devices, including how to safely upgrade the data plane; maintaining consistent behaviors, configuration update and low delay for interactive programming, programming languages, verification of network programs)

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).

As these facets become programmable it is important to develop an understanding of the operational impact of a given datapath program. Some of the questions that will need to be answered are:

- What features are needed, and perhaps as importantly not needed, in a datapath programming language?
- How can one validate that a datapath program meets a set of behavioral requirements?
- How can one validate that a datapath program will operate correctly in the context of an existing network?

- How can one validate the consistency of a set of datapath 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 datapath 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 datapath program changes that protect active traffic and the integrity of the network?

[Haoyu Song] Standard language and interface to support interactive data plane programming

[Kilper] Creating trustworthy hardware for multiple programmable tenants

3. Programmable packet-processing hardware

Text to be written by *who*

[Yan Luo] Creating open research framework(e.g. open chip) for programmable network processing hardware (NPU, FPGA, GPU or any future architectures) through benchmark applications (e.g. NFV), simulation, power modeling, etc..

Infrastructure requirements:

1. Having programmable infrastructure, heterogeneous

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.

[Kilper] White box hardware across multiple emerging technology platforms: sensors, IoT, wireless, optical, and switching implemented at scale, e.g. in a smart city scale deployment

2. Having frameworks, methodologies to allow deployment in SW and HW, that can co-exist with the infrastructure, avoid that programmability is controlled by vendor

Text to be written by *Jerry Sobieski*

[Kilper] Platforms that enable pieces of experimental hardware and software to co-exist together and with commercial hardware and software; applying and removing experimental pieces for periods of time

Marco: Manage programmability across multiple devices: programming languages, verification of network programs, virtualization and slicing of programmable devices, going across different PHYs, maintaining consistent behaviors, deployment into production networks, configuration update and low delay for interactive programming, avoid that programmability is controlled by vendor, discovering of capabilities

Access to infrastructure: Better integration with industry, ability to have early access to technology

<BL> theory of how to safely manage changes to the programmable data plane

<BL> what new capabilities do we need/want now that so many layers (optical, electronic, wireless) can be integrated to a single chip

*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 capabilites. 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 citires 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. An example might be to 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 municiple e-infrastrucutre was ubiquitous and amorphous but could be viewed as a giant intelligent device but programmed in a intuitive manner. Specific hardware should support such partitioning of the component for delegating and insulating.isolating applications elements.

SDI as a Marketplace

This session had 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.

Attendance:

Rodrigo Fonseca (Chair), Tilman Wolf, Cees de Laat, John Wroclawski, John Moore, Munindar P. Singh, Jim Griffioen, Glenn Ricart, Orran Krieger, Bruce Patterson

- **Bruce Patterson: White Paper Response to Call for Participation: SDI/SDXWorkshop**
- **Cees de Laat: The Service Provider Group Framework**
- **James Griioen: 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

Transformative research areas:

1. Economic models and mechanisms

Text to be written by *James / John W.*

1. including of SDI in short timescales/fine granularity/wide area
2. Whether or not the economic mechanisms need to be tied to the technical mechanisms

3. Even the mechanisms for payment should be negotiable

2. Incentives for Cooperation / Governance of Autonomous Entities

Text to be written by *Cees/Munindar*

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 such 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. SDI to the Edge

Text to be written by *Bruce*

New models enabled by programmable infrastructure at the edge could include:
New user models:

- 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

New provider models:

- 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.

In short, slicing to premise edge clearly provides new layers for innovation. More than Internet (as conceived today) available.

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

Depending on specific SDX definition 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. Reliability and manageability (of SDI at the Edge) (can this be merged with 3?)

Text to be written by *who*

5. Service interoperability and composition at SDXes

Text to be written by Glenn

added sets of complications when working with cloudlets you don't own create research opportunities in network-based composition, assembly, and abstraction. (Economic models following the technical mechanisms or not) Dealing with distrust. Norms

Infrastructure requirements:

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

Text to be written by *John M/Bruce(?)/Glenn(?) - Bruce/Glenn: I've taken a stab below, please adjust as required. Thanks, John M*

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.

- Bruce introduces the idea of an SDX connecting individual residences and SMBs

- How do these operate together to deliver services efficiently. What are the economic models?
- How to move (real) money?
- Real users

2. Agents and SImulations

Text to be written by *who*

Interconnecting Across Different Technologies

A summary of the session, to be written by *Mark Berman,*

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 and networking packages on wheels, opening up many new opportunities, but with new constraints arising from safety, privacy, and proprietary technology concerns, among others.

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.

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

[[Raw notes - OK to delete, correct, summarize, etc. - Mark]]

David: SDX as a convergence point for storage, computation, and networking. Creates opportunities for efficiencies beyond what can be done solely with networking. Storage devices already have available computing power, but not easily exposed to applications. With the variety of emerging applications based on Internet, it becomes hard to define end-to-end. We are collecting information that is dynamically generated and hosted in storage devices and make critical decisions based on a subset of information. How SDI can help and facilitate this process?

James Chacko: are there opportunities to chain new FPGA or other hardware-supported services into a NFV-like service chain in SDX? Hardware acceleration is emerging prominently in all levels of computation and self-mutable Hardware, hardware elements that can change their functional capabilities based on real time data, probably help in bringing computation predominantly done on a connected host PC be targeted closer to the-line and thereby improve performance and efficiency in calculating, trimming attacks and even enable extra storage locally.

Madeleine: optical networks offer high-bandwidth, but they have inherent challenges where SDN may be an enabler. Looking at machine learning to gain a better network understanding and adapt. Q: What are the potential opportunities for collaboration between SDN and optical hardware design/implementation? Q: What about protocols designed specifically for analog environments?

Tim: Real soon now, cars will be very sophisticated sensor and networking packages on wheels. How to get benefit relationship between V2V networks and traditional? Car owners have strong privacy concerns - how to address? And, of course, cars are mobile and come on and offline.

Joe: Privacy concerns for participants in a SDX - can policies be de-conflicted without compromising policy privacy.

Chip: Combining Tim's and Joe's talks: potential for lots of data coming from cars - where does it go? How is it disseminated?

Joe: end systems use operating systems to help abstract away some details of heterogeneity.

Mehrdad: SDN control plane scalability for many billions of devices.

Chip: SDX as a meet me point for data

[[End raw notes. -- Mark]]

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. Network (or 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" (ref that section), 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.

Written by *Joe Touch*

Infrastructure requirements:

1. New paradigm 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.

Written by *KC Wang*

2. Need multiple SDXs, with significant data storage and some interesting data sources (sensors, cars) for experimentation.

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 some optical experimentation infrastructure.

Text to be written by *Madeleine Glick and James Chacko.*

We are expecting there to 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

A summary of the session, to be written by *Ethan Katz-Bassett* .

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

Optimizing across layers Aditya/Inder

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.

In the current Internet, each layer of the network builds in complete resiliency and recovery features. Because each layer exposes only a limited interface to the upper layers, keeping

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

Integrating different types of entities and their control loops (disk vs compute vs CDN vs routing) by *Srini/Aditya*

CDN making decisions, ISP doing TE under it

Providing control where it isn't today: to the NIC, to the last mile, etc

- future: QOE in environment with players beyond networks (eyeballs, CDNs, …) and includes business concerns
    - control low level aspects of flows based on QOE goals (CC, routing, QoS, scheduling)
    - beyond control of paths, orchestrating other aspects
- challenges
    - APIs
    - coordination across control planes and players
    - how to model QoE of different users, going from there to control of components

- feedback from QoE observations to knobs you control

The control planes that sophisticated applications, such as video delivery and cloud computing, use to manage computation, storage and communication and those that they use to manage network connectivity currently operate in a completely independent fashion. 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 Open Flow 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, ExSede)

What APIs to allow end-to-end, what control to provide by *Rick*

How to get an abstraction of the pieces and provide the control knobs, then do it at interactive speeds

4. Incorporating business models as part of the interaction

business models, how to incorporate them.

By Ibrahim: 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, ...

(one related point that came up: "separate out technical issues and build living lab, removing all the business etc. business models can follow once there is a clean technical solution")

Infrastructure requirements:

candidates: real users, real traffic

1. Operational infrastructure that exposes rich monitoring

Make it operational and provide monitoring and other information across domains as a basis for end-to-end by Byrav/Aditya?

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. Suitability of use cases/vertical domains--realistic conditions to represent users and services

Realistic conditions: eyeball networks, services, vertical domains/use cases, multiple domains, etc by *Inder/Srini*

- large federated testbed spanning domains (compute, storage, networking, mobile)
    - deep instrumentation to collect data
    - PlanetLab model of distributed ownership/operation
- Existing infrastructure (CloudLab, GENI) missing the interconnection/SDX piece (and doesn't extend out towards client)
    - NSFCloud but bigger
        - can't change dataplane enough
        - program monitoring on the datapath
    - AL2S but more malleable
    - Some of the PEERING (USC/UFMG) work on CLoudLab moves in that direction some
    - Monitoring
        - data path
        - QoE from user

3. Measurement and Control over Lower Layers

integration of layer 1, exposing info and control (with growing data, need to migrate to lower layers, so need to build out controls) by Byrav/Theo

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.

Breakout notes

- future of network: transparent application-controlled bit control
    - separate out technical issues and build living lab, removing all the business etc
        - business models can follow once there is a clean technical solution
- different definitions of end-to-end came up
    - academic / research networks
        - from heart of super computer to storage at remote site
    - stitching QOS from multiple networks at IXPs
    - QOE in environment with players beyond networks (eyeballs, CDNs, …) and includes business concerns
- need performance info, need controller response time

Breakout lightning talks

• Aditya Akella: Toward end-to-end software defined QoE

- motivating trends: programmable network/storage/compute elements, data plane, overlay (CDN, server selection, etc)

- future: control low level aspects of flows based on QOE goals (CC, routing, QoS, scheduling)
    - beyond control of paths, orchestrating other aspects
- challenges
    - APIs
    - coordination across control planes and players
    - how to model QoE of different users, going from there to control of components
        - feedback from QoE observations to knobs you control
- testbeds
    - large federated testbed spanning domains (compute, storage, networking, mobile)
        - deep instrumentation to collect data
    - Existing infrastructure (CloudLab, GENI) missing the interconnection/SDX piece (and doesn't extend out towards client)
        - NSFCloud but bigger
            - can't change dataplane enough
            - program monitoring on the datapath
        - AL2S but more malleable
        - Some of the PEERING (USC/UFMG) work on CLoudLab moves that direction some
        - Monitoring
            - data path
            - QoE from user
- Q: assumption is QoE is in the network? No, not necessarily
- Future networks may look like "content+eyeball stovepipes" that are federated. (For example, Google is building access networks, Comcast is buying lots of content, etc.) These will still need to interconnect...

• Byrav Ramamurthy: Balancing Exchange Points Communication and Inter-controller Communication for Inter-Domain Routing

- many SDXs today talk about improving BGP
- can SDX architecture facility tunnels across domains/controllers?
    - in mobility first, name objects including interdomain tunnels
    - SDX as clearing house (delete/share) such objects?
- end-to-end also includes multiple layers: multi-layer SDN
    - from layer 1 up
- need to incorporate different types of networks
    - R&E networks can't rely on commercial networks for how to operator I2, campus, regional, etc
- east/west interface on SDN controllers not standardized
    - NSI is a standard that is out there, going into production in ESNet, Starlight, GEANT, etc
        - mainly oriented to layer 2
        - API to multiple control planes at exchanges
        - is what exposed to an application same/different from what is exposed to other controller?
            - interface is the same
- goals:
    - advanced TE across domains
- infrastructure
    - facilitate research on future networks
        - include layer 1, layer 2
    - specialized testbeds to attack particular vertical
        - reach out beyond CISE, other industries, society at large
        - health care, vehicular
        - IoT testbed
    - joint ownership/operation/control
        - PlanetLab for SDX

• Inder Monga: End-to-end,multi-domain SDN with SDI bookends

- science DMZ: high speed , high volume campus-to-campus transfer
    - compute platforms at the edge
- each country might have a network, but science is global
- so science networks need protocols for e2e layer2 circuits, allocate BW

- problem: these go from edge of university to edge of university, but application can still suffer from the campus network. how to take from host NIC on one campus to host NIC at other campus?
    - how to automatically configure NIC, LAN, how to coordinate
    - now call campus engineer to configure VLAN from MPLS that ends at edge, drag it across all hops in the campus, someone else configures the NIC. lose all the automation that can happen from campus edge to campus edge
    - SDN techniques to configure from host to host, orchestration and allocation of resources (VLANs, BW, queues, shapers, firewalls, IDS)
    - too many writes going on: if you can coordinate, you can skip many of the writes and skip right to last one
- Q: taking it to the host isn't enough, have to take it to storage system

• Srinivasan Seshan: Cooperation Between Control Planes

- deep dive into video-content distribution
    - delivery handled from video source, broker like Conviva, CDNs with storage/delivery resources, ISPs delivering last mile
    - each has its own optimization and control plane
        - might control different pieces: today Conviva also controls client bitrate decisions
        - each has its own optimization based on its own concerns
        - each player evolved on its own, so you can end up in a local minima
    - QoE joint optimization across all these pieces
        - mirrors optimization of routing across domains, but here players are operating at different levels of control stack
        - similar interactions and monitoring needs to Aditya
    - marketplaces that expose tradeoffs between entities
        - CDN "this is what it would cost to serve from here, this is the performance you would get"
        - some broker can decide
- Q (Theo): user?
    - Also ISPs
- Q: to what degree should the NSF fund research on ISP problems?
    - Y: Industry and academics can't live different worlds, especially now when integrated. If we avoid industry problems, we might work on the wrong problems
    - Problems faced in industry are problems of the Internet as a whole
    - Q: why don't the companies solve their own problems?
    - the Internet has to work for everybody, not just the companies with the most resources
        - take a look at the steam incident
- Q: how does SDI help?
    - If you look at what SDN-like systems are doing, optimizations. So parallel view to end-to-end SDX
- Q: how much is being left on the tables?
    - can point to signs of gross unfairness, resource allocation
    - strategy you would take to build a CDN today would take advantage of inefficiencies in the end-to-end control plane, different than what older CDNs did
- Q: what infrastructure?
    - mimic different CDN strategies: end-user, ISP-hosted. storage/compute distributed throughout something like GENI, not just at cloud DCs
        - GENI racks
- Q: how important is commercial internet connectivity for the testbed?
    - could be beneficial to get real production traffic
    - hook content providers into it: **real live workloads** necessary to evaluate?
    - hard to get commercial providers to buy in
    - real properties without real traffic
- PlanetLab transitioned with CoBlitz, ideas/building blocks for NFV

• Xenofontas Dimitropoulos: Stitching Inter-Domain Paths over IXPs

- Interdomain QOS
- ISPs sell tunnels with service level guarantees within domain

- why hasn't this worked multi-domain?
    - economic issues
        - path broker to stitch across domains
- brokers are taking off in academic networks
- exploit IXPs to stitch interdomain paths under control of path broker
    - ISPs give pathlets with local guarantees
        - BW, maybe latency etc
    - broker stitches end-to-end
    - for critical services
- centrality of IXPs makes them a good point for this stitching
- benefits: path brokers can expose alternative paths, such as via multiple IXPs, that wouldn't be visible in BGP
- small number of large IXPs can cover lots of the address space
- SOSR paper appearing
- scenarios
    - SA client -> local IXP -> AMSIX -> content
    - client -> IXP -> colo
- requirements
    - monitoring
    - ISPs exposing QOS guarantees
- infrastructure
    - SDX that you can control from path broker
    - monitoring
    - management system for path broker
    - manipulate queues in switches

Measuring and Monitoring

A summary of the session, to be written by *Rudra Dutta*

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

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 (left 10h00)
- Dan Kilper
- David Stern
- Nagi Rao
- Rudra Dutta(NCSU)
- Xenofontas Dimitropoulos

Transformative research areas:

1. What and how to measure? How do you know what to measure?

Text to be written by *Nagi*

- 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.
- network-level quantities: latency, bandwidth, jitter - perfSonar-like infrastructure; SDN-related quantities: number of flows; Quantities that may need to be inferred: response time of controller and devices, scalability with respect to number of flows - how fast service is provisioned/restored - function of number and nature of flow entries
- - In various systems, including CPS
  - new "measurement surfaces"
  - how to know what to measure? experiment design, models
  - What is, in essence, the difference in challenges of SDN measurement and SDX measurement?
  - Or SDI? Where the "I" of SDI may be physical (as in CPS - smartgrids, etc.)?
  - Models and analytics (such as used in formal experiment design) are useful
    - Can tell you what you need to measure/pay attention to
    - Without drowning in data or killing the system by measuring

- analytics to extract quantities of interest from measurements
- stability of control systems? real-time need of measuring and transferring

- BUT a big data approach is needed in the background - if only for anomaly detection (like intrusion) - design of experiments

2. What is the right dynamics for measurements?

Text to be written by *Yan (Bala will or might help)*

The 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.

- Things must be programmable, and dynamically configurable
- What should be measured at what frequency, who has authority to determine?
- At what granularities?
- Mechanisms for doing this dynamically?
- In heterogeneous/CPS contexts?

- the line between hardware and software

3. Measuring for accountability - verification when crossing domains

Text to be written by *Dan (Rudra might help)*

- End-to-end is good, but need to know at inter-domain handoff whether guarantees/expectations are met
- Example from optics - when lambda services cross over from one provider to another, how to know who impaired it (if any)?
- Remember many other upcoming PHY - FSO, 60 GHz, THz, just plain today's wireless
- How to make sure measurements are apple-to-apple?
- How to compose these measurements?
- How to analyze? Analytics for from measurements to verifications

4. Information Sharing Incentives

Text to be written by *Rudra*

- Sometimes measurements are hidden where you cannot see them - but somebody else can
- Incentives in terms of non-money tangibles - battery life etc.
- Itself requires measurements to substantiate
- How to provide tailored incentives?
- Need some economists/sociologists in the mix
- Privacy, trust
- Informed consent - what is truly informed?
- Sharing will be automated like measurement itself - must have commonly understood semantics

Infrastructure requirements:

1. Data & M&M Tools Archive (General Methodology, Shareable Approach)

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/experiments.

Text to be written by *Mike*

2. Impose instrumentation need on testbeds

Text to be written by *Mike*

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.

- All testbeds should incorporate instrumentation natively
- Use from above archive if possible
- Otherwise, contribute back to that archive

Participant Notes (In-session):

(If you are in this breakout, please jot down any specific thoughts here - hopefully it will help in the final minutes of this session when we are finalizing the items above. This section is not intended to make it into the final report. Please put your name down with your comments, in case we need to ask questions or discuss.)

Bala's talk: OpenFlaw, because M&M was built in as afterthought, as it often is. Then rearguard action. Suggestion: need library of tools from various players and stakeholders on M&M. David S: we are facing this problem - we want to be able to have monitoring, but be able to turn it on and off. Query-based or specific question-based M&M. Cees said: one word: perfSONAR. Fontas: How is M&M in SDX different from SDNs? Padma seconded that. Marco: OF can only do things that hardware can supply on a common basis. HW should evolve accordingly.

Yan's talk: Measurement has faced several challenges: (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) the diversity and dynamics of measurement tasks: The measurement target could change rapidly, so do the interested metrics; and (3) knowledge discovery from measurement data: the massive measurement data are under-exploited assets that could help learn interesting patterns, predict trends and lead to actions. We need software-defined measurement - programmable M&M. Right place to place this is in the software-defined measurement control plane. Second place: software-defined measurement data 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? Second point: how do we mine measurement data? Across devices? Third point: Based on the above, closing the loop: apply measurement data, learn knowledge and apply to network optimization and control, and if needed, programming new measurement tasks.

David S: OF is the killer app for OF. Padma: this architecture is actually an existing and general one: what is different? Would be good if this was natively integrated. Dan: What you actually have is a cyber-physical system. What you are monitoring may be in the physical part - but your network may go down while trying to transmit monitoring info about a failure. Nagi: with controllers and stuff, new "measurement surfaces" are exposed - we need newer parameters such as controller dynamics in making flow changes, and how fast the new flows realize new/restored state. We do not know how to do that. - measurements and analytics are both needed - Especially with CPS.

Mike's talk: We want to talk about CPS (with real-time aspects, what does M&M mean for real-time?) and about the GENI measurement architecture. In GENI, we measure, but then send data over the management plane, because GENI has one. Question: when things are layered on top of each other, how to measure? Especially if some of those layers are physical. Finally, to characterize a system, you have to be able to do active measurements, install measurement points. summary: with CPS, measurements is getting more complicated. David S: in service chains, we need to be able to know which of five things is bombing out. "measurement" is something different to software and hardware folks. Nagi: Need design of experiment framework for the SDN/SDI domain - need to know how to answer questions on what are measurements that not non-redundant, sufficient, etc. Bala, Padma: back to dynamics issue. Should there be some things you cannot switch off? Dan: yes, in CPSs, say in an automated car.

Ben: we don't have traffic models for a lot of traffic. Can we figure out what you would be even wanting to measure if you wanted to develop a traffic model? This connects to experiment design. Could save us from drowning in data Padma: but you also want anomaly detection.

Dan: How do you measure/verify accountabilty for when transferring domains across technology?

==========

CdL: six points regarding meaningful measurements across domains and across different kinds of resources (not only net's):

- in the future the CI will be managed and controlled by machines, so the information that it needs to control and operate must be machine interpretable. That needs information modelling including the meaning and units of what one is measuring.
- deep learning, machine learning and big data methods will be used to gather and process that information; think of hadoop, etc.
- It is key to research what granularities are needed in measurement. At some point the measurement and metering can become a huge overhead. Need to research 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. P.S. we do not have that in computing except for libraries and api's. look at Multi domain monitoring end to end is what we talk about here. Hybrid resources. Perfsonar future? http://www.perfsonar.net/about/what-is-perfsonar/
- stability of controlled systems?

Monitor and measure also the applications so that you know the effectiveness of the delivered SDI slice.

-

SDXes and the Internet Model

A summary of the session, to be written by *Tom Lehman*

Attendees: Tom Lehman, Arpit Gupta, Ethan Katz-Bassett, Ibrahim Matta, Joe Touch, John Wroclawski, Bruce Patterson, Bryan Larish, David Du, Ken Calvert, Marco Canini, Srinivasan Seshan, Joe Mambretti, Dave Reese

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. Area one - SDX Transformation of Current Exchange Points

SDXes as BGP based valued added system with future vision for no more BGP required

- programming abstractions
- Internet/BGP interoperability without breaking Internet routing
- scalability issues at the local SDX
- Identification of the killer application

Text to be written by *Arpit Gupta*

2. Area two - New Exchange Pont Architectures

SDXs as infrastructure components that allow for programmability, sliceable, hierarchical, multi-layer, and recursive properties.

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.

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 the conventional "slice" model where a process is bound to a single network, and instead 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.

Written by *Ibrahim Matta and Joe Touch*

3. Area three - Distributed SDX Orchestrated Control

Text to be written by *Tom Lehman*

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. Area four - SDX Based Global Communications System Architecture

How an SDN based architecture can evolve to become globally scalable as an any-to-any communications infrastructure, as is the case for IP currently.

Text to be written by *John Wroclawski*

Infrastructure requirements:

1. Requirement one - Testbed that interconnects SDI/SDX prototypes with legacy networks

Provide access to a distributed SDX overlay testbed which operates 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.

Text to be written by *Ethan Katz-Bassett*

2. Requirement two - 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.

Text to be written by *Joe Mambretti, Joe Touch*

Securing SDI, and SDI for Security

A summary of the session, to be written by *Jim Griffioen*

Attendees: Jim Griffioen, Alvaro Cardenas, Rodrigo Fonseca, Munindar Singh, Glenn Ricart, Nick Feamster, James Chacko, Yang Guo, Hongxin Hu, Russ Clark, John Moore, Cees de Laat (10h00 arriving), Malathi Veeraraghavan, Tilman Wolf, Hyojoon Kim, Byrav Ramamurthy, Mehrdad Moradi

Two sub-problems:

- Securing the infrastructure
- Using new technologies in SDI to achieve security guarantees for applications

Transformative research areas:

Proposed Research Topics

- Area 1: Securing SDI/SDX Hardware and Software Infrastructure ("Being Secure") [Nick F.]
    - Responding to attacks
        - Outlining models for attacks against SDI/SDX
        - How do we design a system that allows a distributed SDI system (with distributed trust) react in a coordinated, effective, and trusted way to attacks?
    - Securing infrastructure components
        - How do we secure slices (network traffic, storage, compute).
        - How do we secure the controller (orchestrator) from attack.
        - How do we secure measurement functions?
        - How do we secure the SDI signalling protocols (e.g., secure OpenFlow).
        - Securing physical infrastructure (anything unique to SDI here?)
    - Understanding and minimizing the performance costs of securing the SDI/SDX infrastructure
    - Mechanisms for monitoring and auditing infrastructure security
    - Securing shared resources
        - Policies and mechanisms for authorization and conflict resolution for shared resources
        - What are the right abstractions for security? Ones that simplify the complexity? Is there a common terminology and ontology that are appropriate?
        - How do we translate policy/security requirements into the rules that enforce them (automatically and correctly)? Does machine learning play a role?
    - How do you know that policies that are being defined are secure/correct?
    - How do we define the entities and chains of trust, and responsible parties (liability).
    - What are the security policy languages; can they be automatically verified?

- Area 2: Using SDI to Enhance the Security of Network Applications ("Delivering Security") [Alvaro, Hongxin, Glenn]
    - Providing guarantees (security, compliance, etc.) to applications using the infrastructure
        - State-of-the-art connection tuples based guarantees (OSCARS) has inherent problems - connection tuples (src IP address, dst IP address, src port, dst port) are not tied to an application in advance. There is a need for token-based guarantees. The challenge is that token-based guarantees need changes to transport and routing protocols.
    - Use of slices as a security tool
    - Role of middleboxes or network functions in providing security to applications
        - What are the limitations of traditional hardware-based security middleboxes, such as firewall and IDS, for protecting SDI that envisions a multi-domain and multi-tenant infrastructure?

        - Do we need to redesign existing security middleboxes for programmable and virtualized environments?
    - How do we virtualize security functions using NFV? What should they be? How do we migrate them?
        - Design interoperable, dynamic and reconfigurable network security functions.
        - Elasticity control of network security functions
        - Flexible and on-demon placement of virtual network security functions
        - Address safe migration, correct flow update and optimal provision in virtual security function control
        - Design a high-level service-oriented security language for establishing security service chains and interoperation
        - Provide Security as Service (SaaS).

- Area 3: Social, Legal, Economic, Regulatory Policy, and Trust Challenges ("Shared Security") [Cees, Munindar]
    - Establishing trust relationships across domains in a scalable way
    - Legal and regulatory considerations, enforcement
    - What are the economics of security?
        - How do incentive schemes affect motivations of the parties involved? Specifically, under what conditions and in what ways do different incentive schemes help or hinder security?
        - when money is the incentive
        - when p2p shared use of each other's resource is the incentive
    - How do we author, understand, evaluate appropriate SDI policies?
    - What are privacy implications of the different SDI approaches? How can SDI security promote privacy?

Text for Area 3 [being put together after the session]

This research area deals with the interplay of human and social aspects of security with the technical aspects.

1. Slicing for Security

Text written by *Glenn*

SDI/SDX slicing pioneered by GENI can provide a security encapsulation for sensitive information. The attack surface is decreased 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.

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. Can one slice occupy so many resources it prevents another slice from doing its job? Performance isolation of slices is an independent area for research and is related to the following point on shared infrastructure as well.

2. Authorization and Conflict Resolution Mechanisms for Shared Infrastructure

[Nick Feamster]

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.

3. Exploring the Sociotechnical Design Space for Security

Text to be written by *who*

Munindar's thoughts: Security definitionally is not merely a technical problem. Securing the SDI by itself doesn't make sense we must state securing for whom 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? In fact, that would be the value proposition of software-defined anything. So the question is how to determine the technical policies to be applied. We need ways to capture stakeholder needs effectively as well as making the decision making by the technical elements intelligible to stakeholders.

Individual Attendee Notes

The discussions highlighted a wide range of research challenges that must be addressed to make SDI/SDX systems secure, including:

- security policy description languages,
- secure SDX frameworks/OSes,
- virtualization of security functions,
- transitioning from middleboxs to similar/equivalent security NFVs,
- provision and migration of security functions (elasticity),
- hardening the (single) controller as it is a centralized point of attack,
- understanding the SDI/SDX attack models,
- reconfiguration of SDX security functions on fine grained time scales,
- securing measurement functions,
- enforcing privacy,
- software engineering challenges (e.g., making sure the controller software is secure, robust, reliable, etc), and
- security for mobile SDI systems.

Munindar Singh

Security considerations are inherently challenges of dealing with autonomy. When we take a systems view, it becomes clearer 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.

Tilman Wolf

I think there are two (maybe more) fundamental directions for security research in this area: (1) securing / hardening the mechanisms used in SDI/SDX, (2) ensuring compliances with policies (e.g., access control to resources). For (2), I think there is an interesting problem relating to identity management since identity often is used for access control. Yesterday, several people mentioned the need for complex identity management (federated, etc.). One thing to think about would be to see if we can get away *without* the strong identities. For example, in a marketplace for physical goods, I do not have to reveal my identity - I simply make a cash payment and the merchant gives me the goods. Thus, we could use economic mechanisms to simplify access to SDI (e.g., if you provide payment to cover the cost (or risk of the damage you may do), you can access the resource). Even for "free" access, there could be a deposit that is refunded when the resource is released and nothing bad happened. A related effect is that this approach could also help with privacy. Anyway, my point is that there we should think about mechanisms other than identity for access control.

Byrav Ramamurthy

Securing the software defined infrastructure is a challenging area of research for the community. A hierarchical approach which implements security mechanisms at all layers is essential for securing the SDI. A human-readable set of set of security policies that are automatically translated to machine-readable switch rules will enable scalability and operational simplicity of the SDI. As we transition from a single SDN-controlled domain to a multi-domain framework using SDX, the need to integrate multiple separate security policies and mechanisms becomes apparent. Intra-domain, inter-domain and end-to-end security policy management are key research topics for enabling large-scale SDX deployments for real-world applications. Authorization and access control are also important in a large-scale SDX setting, particularly for operators of one domain to obtain an operational view of another domain in a secure manner for debugging and performance tuning purposes.

Cees de Laat

Trust: In order for SDI end to end security the most important issue is how to create 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, policies 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.

p.s. for trust look at the bank credit cards, I can use worldwide my cards at in principle competing banks. What is there the trust system. (we studied that: <http://delaat.net/sarnet>)

- how can we create trust in a scalable way?
- how do we secure shared resources?
- how to protect the protector, and then off course the protector of the protector ….?

- does slicing make the slices more secure?

Mehrdad Moradi

SDX provides unique opportunities for collaborative security management at exchange points particularly for mobile operators. We know that mobile networks have a few peering points, they have a large volume of signaling traffic, they should support mobile virtual operators. We argue that we should include some fine-grained security policies into peering point contracts for mobile operators. Then, we should design the SDX architecture to handle and enforce these fine-grained policies. For example, the SDX must be able to assure that there will not be any malicious (by any definition) between multiple mobile networks . It also must be able to perform resource-driven security management. For example, there might be some available resources in one mobile network that SDX might be able to leverage to throttle attack traffic of another peer. In general, we argue that we need a richer east-west control and data plane abstractions for performing security management at SDXes. The API must be able to interact with SDI controllers within peers' domain and leverage some compute resources available at the exchange point as well as within neighbouring mobile networks.

James Chacko: How much does the underlying primitives that govern the hardware layer here allow for the introduction of new functions to enable securing a communication. Will these functions be simple rules, ie. will it reside in one plane or need to be homogeneously controlled over both the data and control plane. Securing requires handshakes and lookups and these require communication overhead in a language everyone understands and in policies that are frequently needs updating. How will the introduction of these additional securing techniques affect the latency and efficiency?

Hongxin Hu: In traditional networks, network security functions, such as firewall and IDS/IPS, are generally implemented on vendor proprietary appliances or middleboxes . However, middleboxes usually lack a general programming interface, and their flexibility and versatility are also very limited. On the other hand, traditional hardware-based security functions feature fixed location and capacity, since they are often placed at fixed network entry points and have a constant capacity with respect to the maximum amount of traffic they can process. Such nature of hardware-based security functions renders them awkward in protecting emerging SDI, which enables programmable and virtualized environments. In particular, traditional hardware-based security appliances often work individually with static configuration. However, SDI envisions a multi-domain and multi-tenant Software Defined Exchange (SDX) infrastructure, which requires interoperable , dynamic and reconfigurable network security functions. To address those security challenges introduced by emerging SDI, we need a new security solution, which can virtualizes and utilize network security functions to achieve three goals: (1) Security Elasticity; (2) Security Automation; and (3) Security as Service (SaaS).

Yang Guo: One challenge is how to ensure that (1) the software components in SDN are well designed/implemented and they are secure; and (2) the standard and protocol design takes security into consideration.

Alvaro Cardenas Defining security policies has been one of the fundamental motivations for pushing forward SDN as a great way to manage access control and fine-grained policy enforcement. In traditional SDNs there is a single authority that defines a global security policy for the network. But when we have the interaction of multiple security domains, there is no common language to express, design and verify policies. The key issue in these domains will be (1) Trust (who manages the SDX), and (2) Trust but verify, how can multiple domains check that other domains are conforming to some agreed policy or SLA, (3) privacy-preserving sharing of operations from each of the different domains/authorities for integration into the SDX.

Joon Kim:

One of the important questions in securing the SDI is how to express security policies for multiple domains. Interesting open questions include: How to express security policies, what is the right language, how to come to

agreement with multiple domains, how to enforce, and how to verify that all security policies requirements are met without conflict.

Another interesting research area is about using NFV to protect SDIs, which can be termed as NSFV (Network Security Function Virtualization). Using NFV to attack this problem has certain advantages: easier to make changes, scale up/down, implement special security functions, etc. What are the right mechanisms and language to do are open questions. Weak security in the control channel (controller-switches) is a problem, and a proper secure channel (protocol and specifications) between controllers is missing.

Malathi Veeraraghavan

- Developing formal methods to check for conflicts in security related configurations, e.g., violations in firewall settings, in virtualized systems
- Increased inter-domain collaboration through SDXs and East-West protocols for inter SDN-controller exchanges to improve network security
- Monitoring at network boundaries to check for compliance of security related SLA requirements
- Supporting security functions in high-speed networks, such as those used in R&E networks to support scientific research. Tradeoff between performance requirements and security requirements.

Infrastructure requirements:

1. Translation of physical security policies to virtual infrastructure

Text to be written by *Russ Clark*

2. Both a production infrastructure and an experimental infrastructure are needed.

Text to be written by *Jim Griffioen*

Individual Attendee Notes

Jim Griffioen

Testing and evaluation security challenges requires special purpose infrastructure.

Byrav Ramamurthy

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. In the case of a distributed testbed infrastructure with presence at multiple campuses/locations, it is vital to educate the campus network providers to avoid legitimate experimental traffic from being flagged as a potential cyberattack.

Hongxin Hu

Testing and evaluating security functions in a virtualized environment, which could be an isolated cloud platform.

Russ Clark

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.

Malathi Veeraraghavan

- An infrastructure that enables multiple security services providers to operate on top of basic network infrastructure to promote competition.

Virtualizing SDI in the Datacenter and Cloud

A summary of the session, to be written by *Theo Benson*

Transformative research areas:

1. Multi-scale Cloud (Hierarchy of Clouds)

Text written by *Raj*

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. Multi-domain infrastructure

Text to be written by *Jerry*

how to define (specify, and verify) trust, policy and SLA? What are the right properties to expose to the user?

Applications may not always use 100% of the acquired resources (for example, an application might reserve X Gbps of network bandwidth but not use X Gbps all the time because of a shared resource, say storage/compute, in the end-to-end pipeline). How do we make the unused portion of the resource available to other applications in a transparent way?

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.

Privacy laws & privacy constraints; how to expose regulation constraints and capabilities.

3. Network Function Virtualization

Text to be written by *Andy, Zhi-Li*

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 specification sand 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 application/service performance.

While the flow-based, "match-action" abstraction is adequate for 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 which 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?

4. How to support heterogeneous physical infrastructure in a cloud

Text to be written by *Zhi-Li*

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?

Infrastructure requirements:

1. Open-cloud environments

Text to be written by *Rodrigo*

2. Infrastructure for collecting torrents of data

Text to be written by *Mark/Rick*

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

Text to be written by *Raj/Ilya*

Infrastructure needs to provide storage guarantees/isolation. we need APIs and UIs (hopefully GUIs) that present flexible access to the storage -- this should be backed by infrastructure that supports the different level of flexibility.

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.

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?

Session Template Page

A summary of the session, to be written by *who*

Transformative research areas:

1. Area one

Text to be written by *who*

2. Area two

Text to be written by *who*

3. Area three

Text to be written by *who*

Infrastructure requirements:

1. Requirement one

Text to be written by *who*

2. Requirement two

Text to be written by *who*

Conclusion

To be written by the workshop chairs and volunteers

Appendices

Appendix A: Attendance

To be filled out by Rob

Appendix B: Call for Participation

To be attached by Rob