

Software Defined Networking

Edited by

Pan Hui¹ and Teemu Koponen²

1 TU Berlin, DE, ben@net.t-labs.tu-berlin.de

2 Nicira Networks Inc. – Palo Alto, US, koponen@nicira.com

Abstract

This report documents the talks and discussions of Dagstuhl Seminar 12363 “Software Defined Networking”. The presented talks represent a spectrum of industrial and academic work as well as both technical and organizational developments surrounding Software Defined Networking (SDN). The topic of SDN has garnered significant attention over the past few years in the networking community and beyond, and indeed the term “Software Defined Networking” itself carries different meaning among different circles. A key focus of the talks and discussions presented here is to capture the essence of SDN through concrete network applications, operational experience reports, and open research problems.

Seminar 05.– 08. September, 2012 – www.dagstuhl.de/12363

1998 ACM Subject Classification C.2.1 Network Architecture and Design, C.2.3 Network Operations.

Keywords and phrases Software Defined Networking, Routing, Data centers, Network Abstractions

Digital Object Identifier 10.4230/DagRep.2.9.95

Edited in cooperation with Dan Levin – dan@net.t-labs.tu-berlin.de

1 Executive Summary

Pan Hui

Teemu Koponen

License  Creative Commons BY-NC-ND 3.0 Unported license
© Pan Hui and Teemu Koponen

Software Defined Networks (SDN) is seen as the most promising solution to resolve the challenges in realizing sophisticated network control. SDN builds its promise on the separation of the network control functions from the network switching elements. By moving the control plane out from the network elements into stand-alone servers, the switching elements can remain simple, general-purpose, and cost-effective and at the same time the control plane can rely on design principles of distributed systems in its implementation instead of being confined to distributed routing protocols.

The purpose of the seminar was to look at the current developments in this quickly evolving problem domain and identify future research challenges. The seminar brought together researchers with different domains and backgrounds. Given the high level of interest in SDN from industry, the organizers also invited many participants from companies working with SDN related networking products and services. This mix of people resulted in fruitful discussions and interesting information exchange. The structure of the seminar took advantage of these different backgrounds by focusing on themed talks and group discussions.



Except where otherwise noted, content of this report is licensed under a Creative Commons BY-NC-ND 3.0 Unported license
Software Defined Networking, *Dagstuhl Reports*, Vol. 2, Issue 9, pp. 95–108
Editors: Pan Hui and Teemu Koponen



Dagstuhl Reports

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Organization of the Seminar

Software-Defined Networking (SDN) continues to remain relevant both for the industry and academia and indeed this was very much reflected in the backgrounds of the seminar participants; the seminar had a balanced mix of representatives both from industry and academia.

These two very active communities, industry and academia, are pursuing SDN with different mind-sets, different solutions and different implications in mind, however. The organizers felt that the interactions had been clearly insufficient in the past: practical challenges in SDN continue to remain little known in the academia whereas the industry often remains unaware of the recent useful developments in research. To this end, the two and half day seminar was explicitly structured around this observation; the goal was to allow for fruitful interactions between the industry and academia to maximize the exchange of ideas, challenges and lessons learnt between these two communities.

The seminar discussions and talks were structured around three themes:

1. Status updates. From the very definition to the ongoing standardization work, SDN is still evolving. In these talks and discussions, we dived into the ongoing work at ONF as well as the perceived hard problems to be solved.
2. Industry use cases. In this theme the focus was on exposing the academia to the practical use cases on which industry is working.
3. Implementation. The third theme dived into the details and exposed the seminar participants to both the practical implementation issues faced as well as more theoretical observations about the system design.

For the status updates the seminar had the following talks at the first day. The talks were fairly short so enough discussion could be had between the talks:

- Teemu Koponen: Evolving SDN
- Peter Feil: ONF update
- David Meyer: Hard problems in OF/SDN
- Dirk Kutscher: Northbound interfaces

The discussions after (and during) the talks also bootstrapped the evening and its group discussions about the definition of the SDN and its use cases.

The second day started with the industry use cases.

- Peter Feil: Deutsche Telekom and SDN
- James Kempf: SDN: Definition and Use Cases
- Teemu Koponen: Network virtualization
- Cedric Westphal: SDN for content management/network-based CDN emulation/transparent caching

The rest of the day was dedicated for the implementation theme and a set of short talks were given again to spark the discussion later in the evening about the implementation aspects.

- Dan Levin: State distribution trade-offs in SDN
- Nate Foster: Frenetic
- Toby Moncaster: SDN, can we (IP)FIX it?
- Andrew Moore: S/FPGA/NetFPGA

- Jarno Rajahalme: Issues in routing and tunneling in OF and OVS
- Wolfgang Riedel: Alignment of Storage, Compute and Networking
- Anders Lindgren: Use cases of SDN in information centric mobile networks

The third day was again about the use cases but this time from the academic participants. The following short talks were given with discussions between the talks:

- Christian Rothenberg: RouteFlow
- Fernando Ramos: Secure, trustworthy, resilient SDNs
- Raimo Kantola: Customer Edge Switching
- Frank Dürr: Supporting Communication Middleware with Software-Defined Networking

Outcome of the Seminar

The seminar was well received by the participants. Among the participants there were also organizers of future SDN workshops (IRTF SDN and DIMACS SDN) who signaled the intent of building their workshops around the similar discussion-oriented structure preferred at Dagstuhl.

2 Table of Contents

Executive Summary

<i>Pan Hui and Teemu Koponen</i>	95
--	----

Overview of Talks

Revisiting Routing Control Platforms with the Eyes and Muscles of Software-Defined Networking <i>Christian Esteve Rothenberg</i>	99
Software Defined Networking: overview and use cases <i>James Kempf</i>	99
Evolving SDN: My view on "what the heck is it?" <i>Teemu Koponen</i>	100
Logically Centralized? State Distribution Trade-offs in Software Defined Networks <i>Daniel Levin</i>	100
Hard Problems in Software Defined Networks <i>David Meyer</i>	101
Language-Based Abstractions for SDN <i>Nate Foster</i>	101
Customer Edge Switching <i>Raimo Kantola</i>	102
A Research Overview – “Things that consume my time” <i>Andrew Moore</i>	103
An Update on the Open Networking Foundation <i>Peter Feil</i>	103
Expanding SDN Primitives for Content <i>Cedric Westphal</i>	103
Secure, trustworthy, resilient SDNs <i>Fernando Ramos</i>	104

Panel Discussions





Evening Discussion Day 1	104
Evening Discussion Day 2	105

Participants	108
-------------------------------	-----

3 Overview of Talks

3.1 Revisiting Routing Control Platforms with the Eyes and Muscles of Software-Defined Networking

Christian Esteve Rothenberg (CPqD – Campinas, BR)





License     Creative Commons BY-NC-ND 3.0 Unported license
© Christian Esteve Rothenberg

Joint work of Esteve Rothenberg, Christian; Nascimento, Marcelo; Salvador, Marcos; Correra, Carlos; Lucena, Sidney; Raszuk, Robert

This talk addresses the question on the need and opportunities of combining IP routing protocols in OpenFlow/SDN. The talk aims at raising the debate on (i) transitioning existing networks to OpenFlow/SDN, (ii) hybrid OpenFlow/SDN approaches (i.e. integration with legacy control planes), and (iii) how OpenFlow direct FIB manipulation can help IP routing control applications and enable cost-effective architectures. The research agenda of the RouteFlow project touches prior work on centralized Routing Control Platform (RCP) and its benefits in flexible routing, enhanced security, and ISP connectivity management tasks. This talk calls for input to shape the project research agenda and discusses a number of open research challenges. In addition, we present experiences from prototyping a RouteFlow Control Platform (RFCP) that implements a single node abstraction by means of an AS-wide eBGP routing service.

3.2 Software Defined Networking: overview and use cases


James Kempf (Ericsson – San Jose, US)

License     Creative Commons BY-NC-ND 3.0 Unported license
© James Kempf

This talk presents perspectives and applications of SDN in the context of the wide area network, specifically the mobile core, aggregation of carrier networks as well as within the data-center. Historically, there have been many examples of split-architecture networks enabling the separation of network control from data plane forwarding – following the general principle of policy/mechanism separation. These approaches however did not introduce sufficiently powerful abstractions to realize full convergence of the very diverse set of network and service control interfaces inherent to today’s carrier networks. As carrier networks become more closely integrated with services such as IPTV, customer data hosting, and general infrastructure as a service, there is increasing demand for a unified control plane and management interface for the network. For example, an SDN approach to this problem would coordinate provisioning of storage, processing, and network resources through the same management interface. These use cases depend heavily upon finding the right network and service control abstractions and interfaces. Furthermore, they also introduce complex distributed systems problems of control state management. Nevertheless, SDN presents an opportunity toward unifying today’s complex tangle of network and service control management.

3.3 Evolving SDN: My view on "what the heck is it?"


Teemu Koponen (Nicira Networks Inc. – Palo Alto, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Teemu Koponen

Despite the growing body of research and industrial initiatives based on Software Defined Networks over the past few years, the true essence of SDN remains somehow unclear. This talk shares observations of how today's notion of SDN evolved – from early attempts to improve the manageability of the IP network control plane, to current ideas on better leveraging abstractions in network design and operation. SDN, today emerged from an approach to decouple the control from forwarding, as a mechanism to enable better control (e.g., 4D, Ethane, RCP, Sane). Consequently, OpenFlow and early controller implementations such as NOX emerged as a means to realize this decoupling. The essence of SDN goes beyond the notion of decoupling control from forwarding, however. Once decoupled, modern distributed systems approaches using modular software design principles and tools can be applied to network design and operations. This talk argues that the potential to bring network design, control, and operation into the age of modern software development and deployment, to a large extent, defines the essence of SDN.

3.4 Logically Centralized? State Distribution Trade-offs in Software Defined Networks

Daniel Levin (TU Berlin, DE)


License  Creative Commons BY-NC-ND 3.0 Unported license
© Daniel Levin

Joint work of Levin, Daniel; Wundsam, Andreas; Heller, Brandon; Handigol, Nikhil; Feldmann, Anja

Software Defined Networks (SDN) give network designers freedom to re-factor the network control plane. One core benefit of SDN is that it enables the network control logic to be designed and operated on a global network view, as though it were a centralized application, rather than a distributed system – logically centralized. Regardless of this abstraction, control plane state and logic must inevitably be physically distributed to achieve responsiveness, reliability, and scalability goals. Consequently, we ask: “How does distributed SDN state impact the performance of a logically centralized control application?” Motivated by this question, we characterize the state exchange points in a distributed SDN control plane and identify two key state distribution trade-offs. We simulate these exchange points in the context of an existing SDN load balancer application. We evaluate the impact of inconsistent global network view on load balancer performance and compare different state management approaches. Our results suggest that SDN control state inconsistency significantly degrades performance of logically centralized control applications agnostic to the underlying state distribution.

3.5 Hard Problems in Software Defined Networks


David Meyer (CISCO Systems – Eugene, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© David Meyer

This talk outlines perspectives on the hard research and implementation problems to realizing modern, reliable, robust, deterministic, and practical Software Defined Networks. These problems fall roughly into the categories of technical, social, and economic challenges. Among the technical challenges, the separation of data and control planes introduces scalability and failure-resilience questions as new signalling mechanisms and failure modes may be introduced. Distributed state management of the logically centralized control plane also brings networking further into the realm of distributed and concurrent systems. Achieving the right abstractions for network control that expose enough of the underlying distributed systems complexity to prevent abstraction inversion while still ensuring sufficient control to achieve near optimal network performance goals. Hardware implementations to realize decoupling of the forwarding and control plane face very concrete problems in terms of ASIC design optimization, TCAM space and power trade-offs, etc. Given the right abstractions for network control, reasoning systems toward behavioral correctness guarantees will pose a significant challenge to achieve, but potentially enable new levels of network behavioral determinism. SDN challenges much of the existing dogma in network design and operation, and thus poses a significant sociological challenge to the establishment. SDN challenges or at least motivates revisiting fundamental philosophical design questions: Circuits vs. hop-by-hop forwarding, centralized vs. Distributed control planes, and a shift in influence bases from NetOps to DevOps all must be addressed. Finally, one can argue that the above problems all pose economic challenges to the existing vendor and operator ecosystem. Ultimately, these problems represent a sampling of the difficult problems that SDN presents to us.

3.6 Language-Based Abstractions for SDN

Nate Foster (Cornell University – NY, US)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Nate Foster

Joint work of Guha, Arjun; Gutz Stephen; Harrison, Rob; Monsanto, Chris; Reitblatt, Mark; Rexford, Jennifer; Schlesinger, Cole; Story, Alec; Walker, David

This talk presents examples of how modular software abstraction principles, inspired from the programming languages research domain, can be used to tackle problems in networking. This work leverages a key aspect of Software Defined Networking, the programmable forwarding plane, to enable these abstractions toward realizing more modular, portable, efficient, and understandable networks.


The first principle, component modularity, is frequently used in the software engineering and programming languages domain. Through this principle, system functionality is specified and implemented once, and then reused and composed to realize more complex functionalities. By reducing code duplication, modular composition can reduce system complexity and reduce faults, enabling larger functional modules to be built from smaller, well-understood, and well-tested components. Ideally, network control logic implementation should benefit in the same fashion from the application of modular composition. For example, the forwarding plane state needed to realize the functionality of (1) network topology discovery and (2) network

traffic statistics collection overlap to a degree – that is, both functions share a common subset of instructions. In both functionalities, information about packet arrivals must be collected at the forwarding device and then reported back to a controller. The challenge in enabling these functionalities to be implemented as modular components however lies in how the forwarding state for both functionalities is combined and installed on the devices. The NetCore Language has been developed to this end, to demonstrate how to realize component modularity. High level modular components can be compiled into low-level forwarding instructions to modify forwarding state in order to correctly realize the composition of the modular functional components.

Another example of how software abstraction principles can help ensure efficient and correctly behaving networks, arises from the problem of configuration updates. More specifically, in existing networks, it is extremely hard to reason about the behavior of a network as it transitions from one configuration state to another. Even if a certain property of the network holds before and after the configuration change has been made, e.g. reachability between a given source and destination, it is incredibly difficult to know whether that property will hold at all points through the transition. Abstractions for network update enable network operators to better reason about the behavior of the network throughout the entire transition process. The key idea is that for every packet belonging to a defined flow, that packet must be forwarded at each switch through the network according to one forwarding configuration only. This behavior is realized by keeping both the old and the new forwarding states at every forwarding device within the network. Packets are tagged with an identifier to ensure that as they traverse the network, they are forwarded at each hop according to the same policy, either the old or the new, never a mixture. Consequently, for certain network properties, it becomes possible to guarantee that if the property exists in the starting and ending configuration, that property also exists throughout the configuration transition.

3.7 Customer Edge Switching

Raimo Kantola (Aalto University – Finland)


License  Creative Commons BY-NC-ND 3.0 Unported license
© Raimo Kantola

Joint work of Beijar, Nicklas; Llorente, Jesus; Leppaaho, Petri; Pahlevan, Maryam

This talk describes research efforts toward solving customer edge traffic management and considers the potential for exploiting the mechanisms of the programmable forwarding plane of Software Defined Networks. Customer-facing networks face certain challenges in providing scalable, cost-effective connectivity, while enforcing usage and service level policies and preventing unwanted and malicious traffic from reaching end-users. Customer edge switching (CES) has been proposed as an approach to allow customer-facing network operators to realize these goals, by separating customer connected devices from the public internet and mediating, where possible, connections through application-specific gateways. In theory, by acting as an intermediary between the customer edge network and the public internet, traffic filtering, admission, name resolution and route validity can be more easily established. The mechanisms to realize these behaviors in CES share some semblance with the programmable forwarding plane of Software Defined Networks. While there is ongoing research remaining to understand what aspects of CES may be facilitated with the mechanisms of SDN, proofs of concept demonstrate that certain functionality, i.e., collaborative firewalling, can be achieved.

3.8 A Research Overview – “Things that consume my time”


Andrew Moore (University of Cambridge Computer Lab – Cambridge, UK)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Andrew Moore

This talk broadly documents ongoing group research activities. A primary focus of the group is the NetFPGA, a line-rate, flexible, open networking platform for teaching and research. The NetFPGA is a PCIexpress device that enables researchers to implement among other things, hardware accelerated routers, traffic generators, and OpenFlow packet forwarding. It is an appealing platform to try out new packet-forwarding primitives at line rate. The NetFPGA is but one sub-project of another larger undertaking at Cambridge University, the (MRC²) or Modular Research-based Composably trustworthy Mission-oriented Resilient Clouds project. This project focuses on problems in areas of Data-center switching, distributed resilience, and energy efficiency.

3.9 An Update on the Open Networking Foundation


Peter Feil (Telekom Innovation Laboratories – Berlin, DE)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Peter Feil

This talk provides an overview of the activities and working group structure of the Open Networking Foundation (ONF), with a specific highlight on the organizational goals. The ONF was founded as a non-profit mutual benefit corporation, to promote the development and use of SDN starting with OpenFlow. Members pool their Intellectual Property Rights and may all share within this pool. There exist around 9 current working groups in charge of tasks such as OF specification extensibility, testing interoperability of implementations, marketing and education, and northbound interfaces.

3.10 Expanding SDN Primitives for Content


Cedric Westphal (Huawei US R&D Center – CA, USA)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Cedric Westphal

The current functionality supported by OpenFlow-based software defined networking (SDN) includes switching, routing, tunneling, and some basic fire walling while operating on traffic flows. However, the semantics of SDN do not allow for other operations on the traffic, nor does it allow operations at a higher granularity. In this work, we describe a method to expand the SDN framework to add other network primitives. In particular, we present a method to integrate different network elements (like cache, proxy etc). Here, we focus on storage and caching, but our method could be expanded to other functionality seamlessly. We also present a method to identify content so as to perform per-content policy, as opposed to per flow policy. We have implemented the proposed mechanisms to demonstrate its feasibility.

3.11 Secure, trustworthy, resilient SDNs

Fernando Ramos (University of Lisbon, FCUL, LaSIGE – Lisbon, PT)

License  Creative Commons BY-NC-ND 3.0 Unported license
© Fernando Ramos

This talk presented some of the generic ideas and preliminary research the SDN group in the University of Lisbon (www.navigators.di.fc.ul.pt) is undertaking. It's still all very early-stage, but in summary they are looking at two aspects of Software Defined Networks. First, they focus on the SDN as a distributed system. Their aim is to build secure, resilient, consistent (in its several flavours) distributed control planes. Second, they are using SDN for generic network research and to build robust middle-ware. Examples include improving fault and intrusion tolerant systems, building robust pub/sub middle-ware, connecting massive bioinformatics data processing infrastructures, increasing the efficiency of IPTV networks, among others.

4 Panel Discussions

4.1 Evening Discussion Day 1

```

      \      /      _\/_
      .-'-.      //o\  _\/_
     -- /      \ -- |  /o\
~::~::~::~::~::~====::~::~::~::~::~|~::~::~::~::~
                                     |

```

What's our definition of SDN?

- Applying the same kind of abstractions that we've been using everywhere else in our field of computer science to the field of networking
- Breaking up the vertically integrated, monolithic black boxes to enable more flexibility and innovation based on software concepts
- Relocate control to from tightly integrated devices to "somewhere else"
- Actually, software is an irrelevant part of it. It's abstraction-defined networking and the ability to push the pieces around. The Modularization of it.
- The term software is not just the control – also the ASICs on the forwarding plane.
- The ability for consumers to have vendor-neutral APIs. The thin waist of network management is the CLI, SNMP, and the python expect script.
- Important to consider the semantics of the word "User" is the one who owns the box.

Does it really provide anything new or is it just hype?

- It is nothing new, just packaged up in a nice way that has let it gain traction
- It is a confluence of circumstance
- Routebricks disaster!
- All middle-boxes are x86. Why not everything? Because they're general and bad for everything.

What are the difficult problems ahead? What are the issues that have been definitely solved by now?

- Vendors want to keep control. (One man's opinion)

Is reactive flow processing (i.e., first packet of a flow to controller) essential to SDN?
What's a flow after all?

- Reactivity is not inherent to OpenFlow. Reactivity comes from the need to handle soft-state.
- Difane
- Just think about how an IP router populates its FIB
- There's a hierarchy of life-times for pipes,

Can centralized network control ever scale?

- YES! But it depends on event rates, and where and how you centralize.
- AND it doesn't have to be physically centralized, Logically Centralize!
- We are making separate decisions about distribution of state

What does logical centralization mean?

- When you control the domain, It means you have design choices.
- Eventually Consistency
- Locking
- 2 or 3-Phase Commits
- CAP Theorem
- Transactional
- Which you can achieve depends on the network environment and workload
- It means you can aim for strong consistency models
- You try to slice the functionality of the network to keep separate the divergence
- Ben is not standing next to me right now.

Distributed systems principles are often the magic proposed for scaling, but what are really the mechanisms and principles required to scale?

- See Above
- everything stems from you need for strong consistency.

What about availability, doesn't centralization result in reduced reliability by definition?

- Logical Centralization! See above :-)

There are increasingly many language proposals for SDN. What are the fundamental problems they assist developers with?

- Network dev-ops need ways of specifying network configurations: e.g. the what of the network lookup-algorithms
- How do I specify to my network enforce the following policy and not more
- Is my network actually doing what I think its doing?
- Is the specification correct
- Bluespec
- Network debugging
- There's some real work to be done applying formal methods to reasoning about SDN

SDN is all about making network control more flexible. That tends to result in more complicated network state and systems. What are your thoughts on the implications for testing and debugging?

- By the way: What is a good metric for measuring network control flexibility?
- The total volume of configuration state goes up by orders of magnitude and it becomes too much for a human to reason about
- The abstraction makes it very hard to reason about what actually went wrong

What are the specific tools (or perhaps you have implemented one) that would be helpful?

- OFRewind (maybe. At least the idea)
- FPGAs
- vim and git (it's software!)

Participants

- Bengt Ahlgren
Swedish Institute of Computer
Science – Kista, SE
- Marcus Brunner
Swisscom AG – Bern, CH
- Frank Dürr
Universität Stuttgart, DE
- Lars Eggert
NetApp Deutschland GmbH –
Kirchheim, DE
- Christian Esteve Rothenberg
CPqD – Campinas, BR
- Peter Feil
Deutsche Telekom AG
Laboratories, DE
- Anja Feldmann
TU Berlin, DE
- Nate Foster
Cornell University – Ithaca, US
- Howard Green
Ericsson – San Jose, US
- Pan Hui
T-labs/TU Berlin, DE
- Raimo Kantola
Aalto University, FI
- James Kempf
Ericsson – San Jose, US
- Teemu Koponen
Nicira Networks Inc. –
Palo Alto, US
- Dirk Kutscher
NEC Laboratories Europe –
Heidelberg, DE
- Daniel Levin
TU Berlin, DE
- Anders Lindgren
Swedish Institute of Computer
Science – Kista, SE
- David Meyer
CISCO Systems – Eugene, US
- Toby Moncaster
University of Cambridge, GB
- Andrew W. Moore
University of Cambridge, GB
- Gerd Pflueger
CISCO Systems GmbH –
Halbergmoos, DE
- Jarno Rajahalme
Nokia Siemens Networks –
Espoo, FI
- Wolfgang Riedel
CISCO Systems GmbH –
Halbergmoos, DE
- Sasu Tarkoma
University of Helsinki, FI
- Fernando Manuel Valente
Ramos
University of Lisboa, PT
- Cedric Westphal
Huawei Technologies – Santa
Clara, US

