Achieving Dependability in Software-Defined Networking - A Perspective

Poul E. Heegaard and Bjarne E. Helvik
Department of Telematics
Norwegian University of Science and Technology
Trondheim, Norway
Email: poulh@item.ntnu.no

Veena B. Mendiratta
Network Reliability and Analytics
Bell Labs, Alcatel-Lucent
Naperville, IL, USA
Email: veena.mendiratta@alcatel-lucent.com

***Abstract*—In this paper we take a closer look at the operation of software defined networking (SDN) in intra-domain networks. The focus is on the dependability issues related to interworking of SDN controllers, network OS (NOS), and forwarding in the data plane. Both the separation of the control and data planes, and the (virtually) centralized control processes, are challenging from a dependability perspective. In particular, consistency in operation and information is a challenge, both between the control and data planes, but also within a given plane. To ensure the necessary level of consistency there could be a conflict with the strict real-time requirements given by the per-flow operation of the SDN controller. A principle system model is introduced to discuss the consistency challenge, and to point out undesirable cyclic dependencies between functions that are necessary to configure and operate SDN. The separation of control processing and forwarding do also introduce structural vulnerabilities, which are exemplified.**

***Index Terms*—Software defined networking, dependability, structural analysis, threats**

I. Introduction

Software Defined Networking (SDN), a fast emerging paradigm, is changing how networks are designed and managed. It is claimed that SDN has great potential to change the way networks operate [1], because it has centralized control, which implies simplified algorithms to configure and control sessions on a per flow basis. This has the advantage that you can run the forwarding plane on commodity network hardware, eliminate middleboxes, and ease and open up the network to design and deployment of new third-party service applications.

SDN is an architectural framework for creating programmable networks that are application aware and more open. The two major characteristics are (i) the separation of the control plane from the data plane with the establishment of abstractions between the two planes, and (ii) the consolidation of the control plane whereby a single software control plane controls multiple data plane elements [2]. A high level layered view of the SDN architecture is shown in Figure 1. Thus, SDN enables applications to request and manipulate services provided by the network and allows the network to expose network state back to the applications. The key service offered by SDN is the provisioning of end-to-end services in an optimal manner, and in near real-time where the granularity can vary from individual flows to large traffic aggregates.

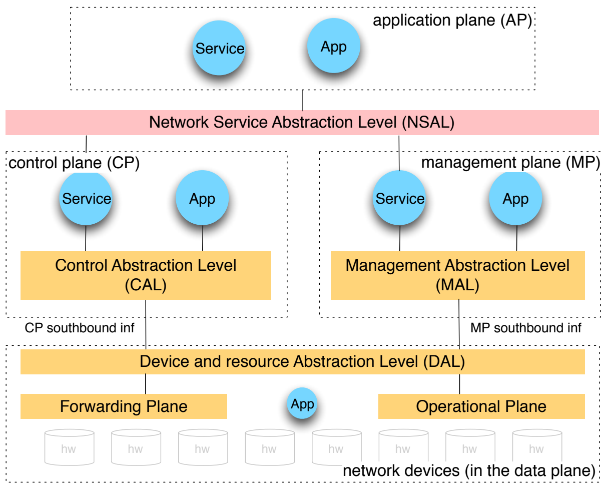
application plane (AP)
Service
App
Network Service Abstraction Level (NSAL)
control plane (CP)
Service
App
Control Abstraction Level (CAL)
management plane (MP)
Service
App
Management Abstraction Level (MAL)
CP southbound inf
MP southbound inf
Device and resource Abstraction Level (DAL)
Forwarding Plane
App
Operational Plane
hw
network devices (in the data plane)

Fig. 1. SDN architecture from IETF RFC7426 [3]

Examples of services provided by SDN include new connectivity services such as bandwidth-on-demand and scheduled bandwidth; multi-layer transport (IP and transport) with cross-layer awareness and convergence; access/aggregation virtualization for differentiation of customer groups with policy enforcement at the network edge, service creation and insertion to automate traffic steering; network analytics and monitoring for optimizing network resources and flows, and load balancing [4]. As seen from Figure 1, management is an integral part of SDN. With respect to dependability, the fault management functionality is of utmost importance.

SDN has attracted a lot of attention over the last few years, and with reports of successful applications of SDN such as in and between Google's data centers [5], the focus is amplified. Two surveys of SDN [1], [6] present the past, present and future of programmable networks starting from the OPENSIG group in 1995 and to SDN today. In addition, the Internet Engineering Task Force (IETF) recently published an RFC 7426 [3] where they define a taxonomy and an architecture for SDN. Both the surveys and the RFC point at several research challenges, but dependability issues are touched on only briefly with respect to the conflict between real-time requirements and the need for consistency.

The main objective of this paper, therefore, is to present a qualitative analysis of the SDN architecture with respect to dependability, in order to provide insights and guidance for the design phase. The main contribution of this paper is the identification of dependability issues that need to be addressed in the design of SDN to ensure proper functionality and dependable operations. A few known dependability issues are recognized, that should either be avoided, or carefully addressed.

In Section II a system model of SDN is described, followed by Section III where different dependability challenges are identified and discussed. Related work is described in Section IV. In Section V, two simple system examples are introduced to demonstrate the potential in structural analysis for assessment of dependability. Section VI contains concluding remarks and directions for further research.

II. System model of the SDN architecture

The data plane does packet forwarding and switching on predefined ports as per the current routing table, which is maintained and updated by the control plane through a routing protocol. In conventional routers the control plane logic and the routing software are highly integrated and deployed on the same hardware node as the forwarding engine. In SDN, the control and the data planes are separated and the complexity of the control logic is moved to an SDN controller. The controller logic is (virtually) centralized, and will typically be deployed on multiple controller processes for fault-tolerance and scalability. The controller processes run on one or more servers that may be co-located or distributed over a geographical distance.

Figure 2 shows a system model of the SDN architecture to point out potential dependability challenges. The model includes the control and data planes in the SDN architecture in Figure 1. The focus is on the interworking of *replicated* SDN controllers, the network OS (NOS), and the forwarding in the data plane through the network devices.

The controller service and application processes in the control plane will be replicated, both for fault-tolerance and load sharing purposes. The best replication strategy is a trade off between the dependability and performance requirements. For fault-tolerance physical and geographical separation is recommended, while for load-sharing it is useful to have shared data disc or memory storage, for which physical, or at least geographical co-location is recommended to minimize propagation delays. The model shows an example of the physical connections between the network devices in the data plane, and (dual-homed) connections to the physical platform of the control plane. This network connectivity is important for the enabling of federation of controller processes, for the geographical separation in the control plane, to get flow state (in the data plane) and to set flow configuration.

III. Dependability challenges

In this section we take a closer look at what can go wrong in terms of failures and data inconsistencies and discuss potential countermeasures to mitigate these problems. Furthermore,

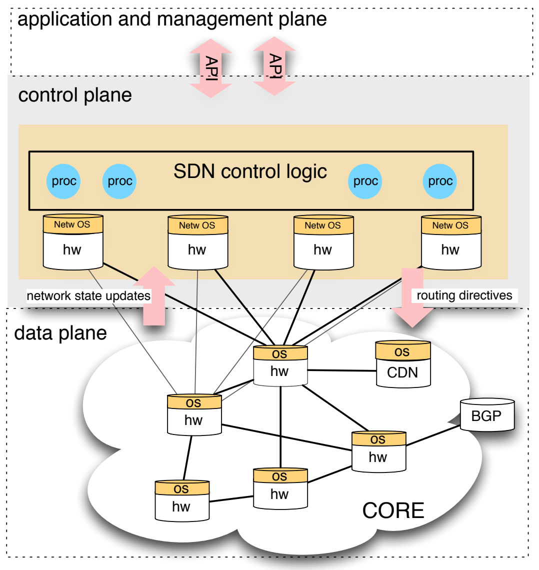

Fig. 2. SDN system model showing the separate data and control planes

we pose fundamental questions regarding the dependability of the proposed SDN architecture in RFC 7426 [3], where the consistency of information is challenging. The necessary means to meet the dependability requirements might be in conflict with the strict real-time requirements given by the per-flow operation of the SDN controller.

A. Services and metrics

The service provided by SDN is the provisioning of end-to-end connectivity for a session or flow on the IP and transport level as per the functional requirements. That is, with the required attributes such as bandwidth, security, multicast or unicast service, QoS, etc.

Availability is the *delivery of service in compliance with the service specification* [7]. Reliability is the *continuity of service in compliance with the service specification* [7].

We define here the availability and reliability metrics in the context of SDN.

- *Availability* - Probability of the readiness to provide service compliant with the requirements, for example, service new demand
- *Partial availability* - Probability of the readiness to provide service compliant with a subset of the requirements, downtime is pro-rated, for example, service some of the demands
- *Reliability* - Probability of the continuity of service compliant with the requirements, for example, provide service for the required duration and then terminate

We have not seen explicit availability and reliability requirements for SDN such as the traditional ”five 9’s” availability requirements for telecom products. A comparable requirement could be ”the SDN control plane should have an availability of x 9s”. Given that the SDN control plane is comprised of multiple federated controllers the expected availability will be greatly influenced not only by the topology and replication scheme, but also by the underlying network and servers used for control plane elements. This indicates the need for availability budgets for each of the elements in the serial path. In addition, availability and reliability requirements can be specified on a per service basis, which would typically be specified in Service Level Agreements (SLAs).

There is a growing concern from the telecom community about the reliability/availability (R/A) of emerging technologies, e.g., Software Defined Networking (SDN), Network Function Virtualization (NFV), Cloud Computing networks, including services provision under failure conditions. Recent discussion in the IEEE/CQR standards groups, and IEEE SRPSDVE Study Group, are suggesting that the reliability requirements of Emerging Technologies (SDN/NFV) should move away from a white-box approach to a black-box approach.

- *White-box approach:* PSTN (public switched telephone network) RAM (Reliability, Availability, Maintenance) metrics or criteria are derived from, and are specific to, PSTN service, architecture and technology. A tight coupling of network functions to Network Elements (NE), architecture and geography exists.
- *Black-box approach:* With emerging network technologies it is not appropriate to transfer Network Function-specific box metrics to multi-service packet NEs using NFV and SDN. In NFV and SDN systems there are multiple applications and multiple services over packet networks, with a diverse set of architectural options and decoupling of control and network functions from the hardware. There is a need for new user-driven R/A metrics based on service criticality, failure modes, costs.

The RAM metrics must be de-coupled from equipment to the Virtual Network Function (VNF). A new metrics hierarchy needs to be established, consisting of service, network, and VNF-subsystem. In a service view, the focus is on user-driven metrics such as service failures, service outage, poor QoE, etc. This requires consistent SLAs that are rich with respect to reliability and availability attributes. In this context, it should be kept in mind that an SDN handling individual flows, defacto is handling virtual paths in the network. Hence, the connection, transfer, disconnect phases, cf. ITU-T I. 350 [8], become relevant in establishing SLAs. The SLAs are derived from the service requirements and will drive the specification of the network requirements.

B. Consistency and availability

Consistency challenges are found between the controller processes in the control plane, between the switches in the data plane, and between the control and data planes. *Horizontal consistency* is between the replicated control processors that need to have the same global view of the current state of the data plane in order to assign new flows optimally while maintaining the existing flows. In the data plane, all switching ports associated with the active flows must be consistently set. *Vertical consistency* means that the state information sent between the data and control plane must (i) reflect the current state of all the switches and flows in the data plane, and (ii) correctly update all ports in the switches of the data plane with the optimal solution from SDN controllers.

In the management plane and in its dealing with the network devices, both horizontal and vertical consistency must be maintained. For management operations, the transaction volume is less and the real time requirements weaker, which eases the maintaining of consistency. However, in handling faults, the management plane relies on a data plane with failed elements to perform a set of consistent operations, which introduces a functional interdependence problem as discussed in Subsection III-F.

For strict guarantees of the consistency atomic actions are required, [9]. These are time consuming, imply a trade-off with availability of the flow forwarding, and are in conflict with the real-time requirements for a flow setup and per incident handling. This will increase the application logic complexity and robustness to inconsistency ([2], and ref therein), and increase the failure probability and consequences of a failure.

The conflict between consistency in data and availability of a function that depends on these data, is a classical problem, e.g., as defined by the CAP-theorem (Consistency, Availability, Partition tolerance) [10]. In SDN the conflict is between strict consistency of the state information about the active flows (both from data plane to the control plane, and between the processes in the control plane) for the optimal flow assignment procedure, and the need for continuity (reliability) of the flow forwarding in the data plane. For short lived flows (”mice”) it is (almost) impossible to have an optimal, consistent flow assignment at any time, which implies that per flow setup might not be feasible. Another, and important, comment is that the optimization of flows should *not reorganize* the global flow assignment for every change in the active flows, because it implies too many, and time consuming, network device reconfigurations. Note also that reorganizations that are not synchronized will cause micro-loops and packet loss.

The bottom line is that the compromise between the consistency and availability must be carefully addressed. The focus should be on potential alternatives to per flow updates (at least for ”mice”-flows), and on the time granularity of the consistency and availability requirements. It is impossible to be ”always” consistent and available. Priority should be given to meet R/A requirements, because typically it is better to ensure that the flows can be setup (system is available), rather than that the flow assignment is always optimal.

The control applications will be plentiful, not necessarily well defined (out of the control of the network operator) and their mixture will change rapidly. This will pose a challenge with respect to the stable operation of the control system.

C. What can fail?

In general, software faults/errors and failures cannot be associated with a single controller node or implementation of a specific functionality, and may have network wide impact (see Table 1):

- Failures dues to design/software.
- "Common mode" among several controllers due to similar software and corresponding failures triggered by the same condition.
- Error propagation; an incorrect internal state is spreading in the system, which is difficult or impossible to rectify by restarts or reloads of individual controller nodes.
- Escalation of failures; the consequence increases throughout (part of) the recovery stages.
- The network OS is likely to be a software dependability "bottleneck".
- Operational/procedural errors (e.g. configuration failures) are the cause of the majority of network outages today. In SDN, the frequency of procedural failures should be reduced because of automation. However, due to the complexity and large span of control, when failures do occur the impact of failures will be widespread with long recovery times, since the root causes are difficult to determine.

D. Network operating system

The network operating system (NOS) in the control plane is responsible for collecting state information from the switching elements, including port assignment, physical topology, active flows. The NOS also updates the ports in each node according to the optimal flow assignment obtained by the optimization process in the SDN controller, which is triggered by new flows, termination of flows, node failure and repair. The update of ports in every affected node has to be an *atomic action* in order to guarantee consistency. This means that an update action must be successfully completed for all switching elements, or not performed at all for any of the elements. Furthermore, no other updates must be active at the same time. Ensuring atomicity is a time-consuming process, that might be in conflict with strict real-time requirements that are put on the update of routing information in the switches on a per flow basis. Many of the flows are short lived. Some of the existing NOS attempt to address these issues. One example is Onyx [11]. Onyx has incorporated work from distributed systems to satisfy the state consistency and durability requirements. It runs an active timer to detect inactive flows when the timer expires. Onyx has a persistent transactional database backed by a replicated state machine for slowly-changing network state and an in-memory distributed hash table (DHT) for fast changing state with weaker consistency requirements. The ONOS system is an open source controller offering similar functionality [12].

There are several concerns that makes the design of a fault-tolerant NOS controller very challenging:

- new fault modes for new virtualization layers
- virtualization separates the HW and the HW fault detection system
- root cause analysis of the network OS failures may be difficult to correlate to the physical layer (HW server failure which caused the NOS failure) and there is a likelihood that new instances of the NOS may be started on the failed HW server
- if not architected properly (NOS and controller) in terms of placement of redundant instances on HW servers, failure of a server could span multiple functions/instances and the probability of a catastrophic failure could be higher than with traditional network physical infrastructure.

E. Path restoration in the data plane

With reconfiguration and rerouting requirements in the order of 10-20 ms, these operations have to be done in hardware [13]. The implication is that the switches need support for this, since neither the OS on the switch, and definitely not the NOS and the SDN controller are able to do this within such a short time window.

F. Functional interdependence

The control system is dependent on the data-plane it is controlling for its functionality, and vice versa. Setting up a *depends on graph* [14] between the (internal) services in the system may yield a graph with cycles, where fault tolerance and proper fault handling cannot be guaranteed.

In a *depends on graph*, cycles of dependencies should not exist. Figure 3 shows the relation between management of path setup in the control plane, which depends on that the paths in the data plane are correct. This is only possible if the path is correctly selected and setup. The cyclic dependency is bad from the dependability perspective. However, as shown in Figure 3, a solution is to pre-provision redundant paths in the data plane dedicated for control traffic. This includes the updates of data plane state information, and the configuration of the new paths in the data plane. The cycle is then broken.

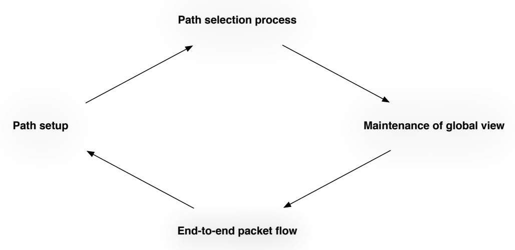

Fig. 3. The depend-on-graph for SDN. With preplanning of redundant paths in the data plane for control traffic, the inter-dependency is removed as indicated by the crossed out link. Then the dependency cycles is broken.

G. SDN in the inter-domain

It is not clear how SDN will work in the inter-domain, and how the interworking will be between the conventional IP domain and an SDN enabled domain. There is a minimum of information that needs to be exchanged between domain-specific control planes to optimize and improve the end-to-end performance and dependability, and also security, e.g. detecting attacks in progress. Proper incentives must be

TABLE I
THREATS TO DEPENDABILITY

Error in	Failure impact
1. Controller	
• NOS instances	Restart NOS instance
	• partial outage for restart duration
	• includes time to get state information from FE
• Controller software	Recovery based on redundancy scheme implemented
	• loss of service in transition state (e.g., connection setup)
	• loss of state information if cold standby or no checkpointing
	• partial outage for recovery duration
• Controller hardware	Recovery based on redundancy scheme implemented
	• loss of state information if cold standby or no checkpointing
	• partial outage for recovery duration
2. Connectivity	
• Link	NOS instance (incorrectly) assumes FE is failed and updates global view accordingly, which results in inconsistent state information among NOS instances until the link failure is detected and the physical network automatically reconfigures (5 ms in the best case)
• Forwarding element (FE)	Inconsistent global network view for the duration of the error detection interval for the FE failure
• NOS instance and FE	Incorrect state information in NOS instance
3. Between controller instances	
• Intra-domain	Inconsistent state information amongst controllers, which implies service failure due to incorrect global network view, partial downtime confined to impact controller instances, and in worst case is total downtime
• Inter-domain	
4. Design or software	Likely to have network wide impact
• Common mode software	Failure of several controllers
• Incorrect internal state	Error propagation of incorrect state, requires system restart and not only restart of individual nodes
• Failure escalation	
5. Procedural	
• Mis-configuration of networks	

defined for the providers and operators to exchange more information, for instance though a richer SLA. Furthermore, the interworking between communication systems, and other critical infrastructures, such as the Smart Grid, is an unsolved challenge. In this interworking, we not only increase the complexity of the (virtually centralized) control, but also add logic and increase the complexity to the controlled devices.

IV. Related Work

Given the great interest in SDN there has been considerable work published on the topic in recent years. However, work in terms of the overall system dependability of SDN is rather limited. Different studies focus on either the controller logic or the network OS. In this section we survey the existing work on SDN dependability and identify where there are gaps.

Kreutze et al. [6] provide a comprehensive survey on SDN covering its context, rationale, main concepts, distinctive features, and future challenges; and include the research efforts and challenges for the design of switches and control platforms with a focus on resiliency, scalability, performance, security, and dependability. In a subsequent work they [15] posit that security and dependability of the SDN is still an open issue and list threat vectors that can enable the exploitation of SDN vulnerabilities, and outline concepts for the design of a secure and dependable SDN control platform.

The reliability and scalability issues for SDN that are presented in [16], focus on additional computational and network resources consumed due to the decoupling of control and data planes which can lead to additional failures. They study disaster scenarios running experiments on a GENI test-bed.

Algorithms for the reliability-aware placement of controllers in SDN are developed to maximize the reliability of the control network [17]. The metric used to characterize the control network reliability is the expected percentage of control path loss. Their results show that proper placement of controllers in the SDN can improve reliability without introducing unacceptable switch-to-controller latencies.

Sharma et al. [13] focus on fault tolerance of OpenFlow (a communications protocol used in SDN to access the data plane) for deployment in carrier-grade networks where there is a requirement that the network recovers from switch and link failures within a 50 ms interval. Their simulation results show that if the controller has to notify all the switches about recovery actions (restoration and protection) this can cause a significant load on the controller and OpenFlow may not be able to achieve failure recovery within a 50 ms interval. Adding the recovery action in the switches then the switches can do recovery without the controller and achieve recovery within 50 ms in a large-scale network serving many flows.

Shalimov et al. [18] present an analysis of the performance, scalability, reliability, and security of some open source SDN/OpenFlow controllers (NOX, POX, Beacon, Floodlight, MuL, Maestro, Ryu). Their overall results indicate that the tested controllers are not ready to be used in production, and have to be improved in order to improve all the above mentioned characteristics. Reliability here is the *ability of the controller to continuously provide service*, which means to provide service under an average workload without accidentally closing connections with switches or dropping OpenFlow messages from the switches. To evaluate the reliability, the number of failures during long-term testing under a given workload is measured.

The work by Ros et al. [19] is focused on determining how many controllers need to be instantiated, where they must be deployed, and what network nodes are under control of each of them, in order to achieve at least five nines reliability in the southbound interface between controllers and nodes. For this, the Fault Tolerant Controller Placement problem is presented and a heuristic algorithm is developed that computes placements with the required reliability. The algorithm is run on a set of 124 publicly available network topologies. The results indicate that each node is required to connect to just 2 or 3 controllers, which typically provides more than five nines reliability. At the same time, the total number of controllers varies greatly and is more related to the network topology than to the network size, 10 controllers or less cover 75 % of the most interesting cases.

To achieve resilient control traffic forwarding, the authors of [20] investigate the protection of control traffic in SDNs with multiple controllers combining local rerouting and constrained reverse forwarding protection. This scheme enables switches to locally react to failures and redirect the control traffic to controllers by using standby backup forwarding options. The goal is to find a set of primary routes for control traffic whereas much control traffic as possible can benefit from the proposed protection scheme. Simulation results on real topologies show that the approach significantly improves the resilience of control traffic.

In SDN the network controller can be a single point of failure. In [21] the authors consider different active/standby strategies to provide a controller failover in the event of a failure. A high-availability controller architecture is proposed and a prototype is developed to demonstrate the efficiency of the solution and demonstrate experimental results.

V. Examples: Structural Analysis of SDN

In this section two examples of qualitative structural analysis are given. The examples illustrate the increase in the vulnerability of an SDN system, which requires connectivity between network elements and the controller(s) in addition to connectivity in the forwarding path.

A. Simple example

In the first example we are considering the SDN system model from Section II. The four replicated control processes are merged into one highly reliable process, which is dual-homed with node 2 and 4. The CDN and BGP nodes are not included in the model. This is illustrated in Figure 2 where also the network elements (links and nodes) are labelled.

From a qualitative assessment perspective, the vulnerability is best illustrated by identifying the *minimal cut sets*, S, i.e., the minimal sets of network elements whose failure causes inability to establish a connection between node n_1 and n_5, [22]. We assume that both nodes and links in the system may fail.

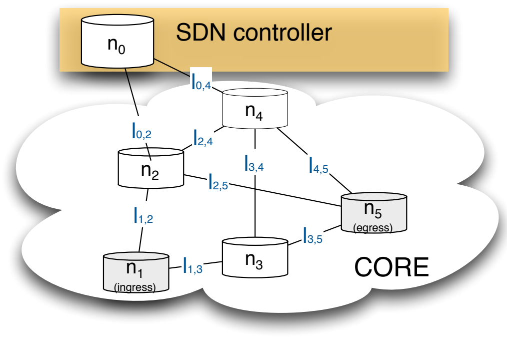

Fig. 4. Case for structural analysis of SDN model in Figure 2

Establishing a connection requires the following paths:

- *flow triggering*: a path for the trigger message that should be sent from n_1 to n_0 (SDN controller) on arrival of a new flow
- *network state update and route directives*: a path from the SDN controller to each node, n_i, $i = 1, \cdots, 5$.
- *forwarding*: a path from n_1 to n_5

The structural analysis for the connection from n_1 to n_5 in the SDN example, identifies that S consists of 26 minimal cut sets, $S = \{s_1, \ldots, s_{26}\}$:

$$\begin{aligned} S \;=\; & \{n_0, n_1, \{n_2, n_3\}, \{n_2, n_4\}, \{n_2, l_{0,4}\}, \{n_2, l_{1,3}\}, \\ & \{n_2, l_{3,4}, l_{3,5}\}, \{n_2, l_{3,4}, l_{4,5}\}, \{n_2, l_{3,5}, l_{4,5}\}, \{n_3, n_4, l_{2,5}\}, \\ & \{n_3, l_{1,2}\}, \{n_3, l_{2,4}, l_{2,5}\}, \{n_3, l_{2,5}, l_{4,5}\}, \{n_4, l_{0,2}\}, \\ & \{n_4, l_{1,2}, l_{2,5}\}, \{n_4, l_{1,2}, l_{3,5}\}, \{n_4, l_{1,3}, l_{2,5}\}, \\ & \{n_4, l_{2,5}, l_{3,5}\}, n_5, \{l_{1,2}, l_{1,3}\}, \{l_{1,2}, l_{2,4}, l_{3,5}, l_{4,5}\}, \\ & \{l_{1,2}, l_{3,4}, l_{3,5}\}, \{l_{1,3}, l_{2,4}, l_{2,5}\}, \{l_{1,3}, l_{2,5}, l_{3,4}, l_{4,5}\}, \\ & \{l_{2,4}, l_{2,5}, l_{3,4}, l_{3,5}\}, \{l_{2,5}, l_{3,5}, l_{4,5}\}\} \end{aligned}$$

The vulnerability in this example study is reflected in the number of low cardinality cut sets, i.e., the system fails with few simultaneous network element failures. Hence, we regard the cardinality, $c_j = \|s_j\|$ of each of the minimal cut sets, $j = 1, \cdots 26$. In Table II each column contains the number of sets that have cardinality k, i.e., $C_k = \|\{s_j \in S | c_j = k\}\|$, $k = 1, 2, 3, 4$. The table compares the minimal cut sets of an SDN system with a conventional IP network where the control plane is embedded in the nodes, n_i, $i = 1, \cdots, 5$. For the conventional IP network control node n_0 is not required and only a forwarding path is needed. From the table we can see that the number of minimal cut sets with cardinality one has increased from 2 (the peering nodes n_1 and n_5) to 3 due to the inclusion of control node n_0. Furthermore, observe that the number of minimal cut sets with cardinality 2 has increased from 4 to 7. This indicates that, even in this very simple

TABLE II
THE DISTRIBUTION OF THE CARDINALITY OF THE MINIMUM CUT SETS FOR AN IP NETWORK AND AN SDN

	C_1	C_2	C_3	C_4	sum
IP network	2	4	12	3	21
SDN	3	7	13	3	26

example, a significant increase in vulnerability is observed for the SDN case that is not explained solely by the introduction of a single control node. As the next example shows, similar results are obtained for larger networks with two independent controller sites that are disjoint and dual-homed to the network.

B. A nation-wide backbone network example

In the second example we study a nation-wide backbone network that consists of 10 nodes across 4 cities, and two dual-homed SDN controllers, see Figure 5 for an illustration of the topology. The nodes are located in the four major cities in Norway, Bergen (BRG), Trondheim (TRD), Stavanger (STV), and Oslo (OSL). Each town has duplicated nodes, except Oslo which has four nodes (OSL1 and OSL2). The duplicated nodes are labelled, X_1 and X_2, where X=OSL1, OSL2, BRG, STV, and TRD. In addition to the forwarding nodes, there are two dual-homed SDN controllers (SC$_1$ and SC$_2$), which are connected to TRD and OSL1.

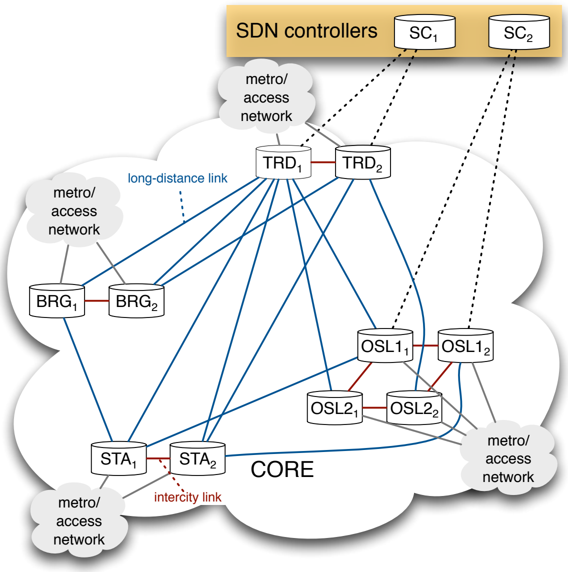

Fig. 5. Case for structural analysis of SDN model of a nation-wide backbone network

The objective of the study is again to compare SDN with traditional IP network, with the same topology and localization of IP routers and SDN forwarding switches. We assume that nodes, links and controllers in the system may fail. The peering traffic in a city is routed through an access and metro network, with a connection to both (all four) nodes in the city. The system is working, i.e., is up, when all the access and metro networks are connected. Note that for SDN, a controller must also be reachable from all nodes along a working path.

The structural analysis for all the possible connections in the SDN example shows that the cardinality of the set of minimum cut sets, S, is $\|S\| = 2916$. The cardinality, $c_j = \|s_j\|$ of each of the minimal cut sets, $j = 1, \cdots 2916$ is given in Table III. Each column contains the number of sets that is $C_k = \|\{s_j \in S | c_j = k\}\|$, $k = 1, \cdots, 13$. The table compares the minimal cut sets of SDN with a conventional IP network where the control plane is embedded in the nodes, and hence, no controllers are needed.

The number of minimal cut sets with cardinality one is equal to zero because traffic sources are at least dual-homed and there are two dual-homed control sites. The number of C_2 minimal cut sets has increased from 3 to 4 due to the control nodes. Note that also the number of C_3 minimal cut sets has almost doubled. This indicates that in this example, a significant increase in vulnerability is observed for the SDN case that is not explained solely by the introduction of a control node, but the fact that a controller must be reachable from every node across the backbone in order for the network to be working.

C. Two level hierarchical modeling

From the structural analysis in this section, assuming independence between network elements, the network availability may be obtained by applying the inclusion-exclusion principle to minimal cut sets, provided that the availability of all the network elements is known. There are two main challenges with this approach (i) the network elements are not always independent with respect to failure and repair, (ii) to obtain the availability of the network elements is challenging as elements are comprised of hardware and software components (sub-elements) with many failure modes and repair strategies. With more detailed models of the network elements, where the compound elements are subdivided into smaller, and to some extent known components, it is more likely that we can estimate the model parameters, and come up with reasonable numerical values that can be applied for quantitative assessment. Such models can be expressed applying for instance Markov models or (Stochastic) Petri Nets. Hence, a hierarchical combination of structural models to handle the large scale of a full network, and more detailed model for each type of elements, or a group of interdependent elements, is regarded as a viable approach. In [23] an extension of the system example of this section is presented, where such a two level modelling approach is applied.

VI. CLOSING REMARKS

This paper presents some of the dependability challenges associated with the operation of SDN networks. There are several areas of concern, one being that the controlling of the forwarding plane depends on the very same forwarding plane that is to be controlled, and another is the potential side-effects of moving from traditional networking with autonomous and distributed control designed for connectivity, to a centralized

TABLE III
THE DISTRIBUTION OF CARDINALITY OF THE MINIMUM CUT SETS FOR THE IP NETWORK AND SDN

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}	C_{12}	C_{13}	sum
IP network	0	3	8	91	304	360	356	189	70	13				1394
SDN	0	4	15	107	340	520	780	584	302	170	59	31	4	2916

control with focus on QoS and cross-layer resource utilization. The dependability challenges related to this include how to ensure consistency in operation and information both between the control and data planes, but also within each of the control and data planes. The *depends on graph* illustrates this with a principal system model and points out undesirable cyclic dependencies between functions that are necessary to configure and operate SDN properly. There is a tradeoff between providing continuous guaranteed consistency and the real-time requirements given by the failure handling, and the path setup. Another tradeoff is the need to geographical separation of controller process replication for fault tolerance, versus the co-location of the same replicas to enables efficient load-sharing.

Two examples of simple structural analysis of SDN are presented, which show that the minimal number of cut sets with a given cardinality is greater for SDN than for a comparable IP network. To demonstrate the full potential of the qualitative assessment approach, even larger examples should be studied. Furthermore, for quantitative assessment a hierarchical combination of structural models to handle the large scale of a full network, and more detailed dynamic model for each type of elements, or a group of interdependent elements, is regarded as a viable approach. This is work in progress that will make a more realistic case and the improve the relevance of the analysis.

In order for the SDN to be applicable in the inter-domain, and in interworking between the conventional IP domain and an SDN enabled domain, there is a minimum of information that must be exchanged to manage the end-to-end security, performance and dependability. An open and important research question is how to define proper incentives so that the providers and operators are willing to exchange more information, for instance through richer SLAs.

REFERENCES

[1] B. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks," *Communications Surveys Tutorials, IEEE*, vol. 16, no. 3, pp. 1617–1634, Third 2014. [Online]. Available: http://doi.acm.org/10.1109/SURV.2014.012214.00180

[2] N. Feamster, J. Rexford, and E. Zegura, "The road to SDN: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[3] E. H. (Ed.), K. P. (Ed.), S. Denazis, J. H. Salim, and O. Koufopavlou, "IRTF RFC 7426: Software-Defined Networking (SDN): Layers and Architecture Terminology," Internet Research Task Force (IRTF), RFC 7046, Jan 2015. [Online]. Available: http://tools.ietf.org/pdf/rfc7426.pdf

[4] "Gaining full control of your network with service provider SDN," Alcatel-Lucent, Strategic White Paper, 20i4. [Online]. Available: http://www.alcatel-lucent.com/solutions/software-defined-networking

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hlzle, S. Stuart, and A. Vahdat, "B4: Experience with a globally-deployed software defined wan," in *Proceedings from SIGCOMM 2013*. Hong Kong, China: ACM, 2013.

[6] D. Kreutz, F. M. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[7] A. Avizienis, J. C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, pp. 11–33, 2004.

[8] ITU-T I.350 (03/1993), "General aspects of quality of service and network performance in digital networks, including ISDNs." [Online]. Available: http://www.itu.int/rec/T-REC-I.350-199303-I

[9] S. Mullender, Ed., *Distributed Systems*, 2nd ed. Addison-Wesley, 1993.

[10] E. A. Brewer, "Towards robust distributed systems," in *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, ser. PODC '00. New York, NY, USA: ACM, 2000. [Online]. Available: http://doi.acm.org/10.1145/343477.343502

[11] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: A distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924943.1924968

[12] ON.Lab.ONOS, "Open network operating system." [Online]. Available: http://tinyurl.com/pjs9eyw

[13] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "OpenFlow: Meeting carrier-grade recovery requirements," *Computer Communications*, vol. 36, no. 6, pp. 656 – 665, 2013, reliable Network-based Services. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2012.09.011

[14] F. Cristian, B. Dancey, and J. Dehn, "Fault-tolerance in the advanced automation system," in *Fault-Tolerant Computing, 1990. FTCS-20. Digest of Papers., 20th International Symposium*, June 1990, pp. 6–17.

[15] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 55–60.

[16] X. Guan, B.-Y. Choi, and S. Song, "Reliability and scalability issues in software defined network frameworks," in *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*. IEEE, 2013, pp. 102–103.

[17] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On reliability-optimized controller placement for software-defined networks," *Communications, China*, vol. 11, no. 2, pp. 38–54, 2014.

[18] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, "Advanced study of SDN/OpenFlow controllers," in *Proceedings of the 9th Central &#38; Eastern European Software Engineering Conference in Russia*, ser. CEE-SECR '13. New York, NY, USA: ACM, 2013, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2556610.2556621

[19] F. J. Ros and P. M. Ruiz, "Five nines of southbound reliability in software-defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 31–36.

[20] Y. Hu, W. Wendong, G. Xiangyang, C. H. Liu, X. Que, and S. Cheng, "Control traffic protection in software-defined networks," in *Global Communications Conference (GLOBECOM), 2014 IEEE*. IEEE, 2014, pp. 1878–1883.

[21] V. Pashkov, A. Shalimov, and R. Smeliansky, "Controller failover for SDN enterprise networks," in *Science and Technology Conference (Modern Networking Technologies)(MoNeTeC), 2014 First International*. IEEE, 2014, pp. 1–6.

[22] R. E. Barlow and F. Proschan, *Statistical Theory of Reliability and Life Testing: Probability Models*. Holt, Rinehart and Winston, 1975.

[23] P. E. Heegaard, B. E. Helvik, G. Nencioni, and J. Wäfler, "Managed dependability in interacting systems," Work in progress.