2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)

Distributed Software Defined Networking Controller Failure Mode and Availability Analysis

Paul Reeser
Advanced Technology & Systems
AT&T, Inc.
Middletown, USA
preeser@att.com

Guilhem Tesseyre
Cloud Solution Group
Juniper Networks, Inc.
Sunnyvale, USA
gtesseyre@juniper.net

Marcus Callaway
Technology Service & Operations
AT&T, Inc.
Atlanta, USA
mc8126@att.com

***Abstract*— Given the critical role Software Defined Networking controllers play in cloud computing and networking architectures, understanding their resiliency profile is crucial. Using OpenContrail as a reference architecture, we analyze the typical distributed controller failure modes and their effects on the control and data planes. We then develop hardware- and software-centric theoretical availability models for a variety of physical topologies and software modes of operation. These parametric models are used to predict availability and quantify sensitivity to underlying platform and process resiliency. The results suggest that the distributed control plane can achieve very high availability, while the host data plane may achieve much lower availability due to inherent single points of failure.**

Keywords- SDN, NFV, FMEA, Availability, Modeling

I. INTRODUCTION

OpenContrail [1] is one of many Open Source and vendor extensible overlay platforms for Software Defined Networking (SDN) within and between cloud environments, including ODL [2], ONOS [3], Floodlight [4], NSX [5], etc. OpenContrail consists of two main components: the Controller and the vRouter. The Controller is a logically centralized but physically distributed SDN controller responsible for providing the configuration, control, and analytics functions of the subtending virtualized network. The vRouter (similar to the Open vSwitch [6]) is the forwarding plane of a distributed router that runs in the hypervisor of a virtualized server, and extends the network from the physical routers and switches in a data center into a virtual overlay network hosted in the virtualized servers. The OpenContrail Controller provides the logically centralized management or *control plane*, and orchestrates the host vRouter *data planes*.

From a high availability standpoint, one of the compelling benefits of distributed SDN controllers such as OpenContrail is that the logically centralized Controller can be highly physically distributed. As described in section II, the Controller consists of multiple node types, each of which has multiple instances for increased availability and horizontal scaling. Those node instances can be physical servers or virtual machines (VMs). Multiple node types can be combined on a single server or spread across multiple physical servers.

Given the critical role that SDN controllers play in cloud computing and networking, understanding their resiliency profile is crucial. In this work, we analyze OpenContrail 3.x process failures modes and their effects on the Controller control plane and the subtending host vRouter data planes within a cloud environment (data center). We then develop hardware- and software-centric theoretical availability models for a wide variety of physical hardware (HW) deployment topologies and software (SW) modes of operation. These parametric models are used to predict availability and quantify sensitivity to underlying HW and process resiliency.

Although OpenContrail is used as an example, our goal is to develop a flexible, extensible analytic modeling framework that can be used to assess the relative availability of any distributed SDN controller in a variety of HW configurations. From a Controller SW standpoint, we fully encapsulate the implementation (OpenContrail 3.x in this case) into two tables (II and III) and develop the model framework around these inputs so that other implementations can be analyzed simply by populating these two tables appropriately. The parametric model framework is thus highly flexible and easily adaptable to other distributed SDN controller node architectures. From a HW deployment standpoint, we consider topologies spanning the extremes from the smallest reasonable scenario (minimum number of VMs required to support the quorum, each on a single host, all in a single rack) to the largest necessary scenario (every role copy in its own VM, each on its own host, distributed over a quorum number of racks).

Much prior work has been done to quantify the impact of HW failures and SW faults on forwarding plane availability in SDN-controlled networks [7-13]. These works model each SDN controller data center as a single entity, and focus on availability optimization at the network of controllers level. As an example, [9] develops an overall availability model of the high-level network of links, switches, routers, and controllers comprising an SDN-controlled backbone network. The SDN controller node itself is modeled as a 'K of M' cluster of identical HW/SW elements. In these works, 'distributed' means that SDN control is spread across multiple controller nodes in a backbone network of data centers.

In contrast, the focus of this work is to decompose the SDN controller node itself into the various roles (network routing configuration, forwarding plane control, data collection and analytics, etc.). We then drill down to the controller process level, capturing such issues as availability critical path process counts, inter-process dependencies, and quorum requirements. Here 'distributed' means that the intra-data center SDN controller SW roles are physically spread across a variety of HW layout and SW container topologies. To our knowledge, this analysis is the first effort of its kind.

In section II we summarize the OpenContrail 3.x SW architecture, and in section III we describe its process failure modes and resiliency impacts. Next, in section IV we propose various reference HW deployment topologies as a concrete framework for the subsequent analytic modeling. Then, in section V we develop HW-centric parametric availability models to analyze the impact of physical deployment topology, and in section VI we develop SW-centric parametric availability models to analyze the impact of process modes of operation. Finally, section VII provides concluding remarks.

II. OpenContrail Architecture

OpenContrail is segmented into several functions from an architectural perspective, called node types or roles. This segmentation is intended to leverage the resiliency capabilities of a scalable architecture of loosely coupled services.

The *Config* node provides a means for clients to consume services by transforming a high-level networking request into a lower level object. This object is then passed to the Control node, and ultimately to the vRouter to enforce network policies. A northbound API server exposes configurable objects to install configuration state and perform create-read-update-delete operations. The API process is responsible for storing/loading permanent data to/from the database (DB). A Cassandra cluster is used to persistently store config data, and a Zookeeper ensemble is used to guarantee uniqueness of system-generated IDs. A schema transformer is implemented to translate the high-level data into the low-level object. An Interface for Metadata Access Points (IF-MAP) server provides a mechanism to write the transformed low-level data, and a southbound interface to push this data to the Control nodes. RabbitMQ is used as a messaging bus for communication between the processes. The discovery server is used to locate other nodes providing a particular service.

The *Control* node (IF-MAP client) retrieves information from the Config node and passes the data to vRouter agents, gateways, and other Control nodes. All Control nodes are active, and synchronize their routing information using Border Gateway Protocol (BGP). The vRouter agents running on the Compute hosts connect to two Control nodes simultaneously, and download their configuration and routing information using this established communication channel.

The *Analytics* node collects and exposes operational data for the cluster, including logs, stats, queries, alarms, and event streams. A Cassandra cluster is used to store Analytics data. Data generators (processes, and physical or virtual devices) send their data to the collector, which then uses a Redis cache to store real-time data or a Cassandra DB to store persistent data. A Kafka bus is also used to stream events and alarms.

The *Database* node uses Cassandra for both the Config and Analytics roles. Separate Cassandra DBs are used for the Config and Analytics data. In addition, the Config role uses Zookeeper, and the Analytics role uses Kafka. Each of these four components is clustered in a 2N+1 fashion, where N is the number of failures supported. We assume that N=1; that is, we consider the minimum 3 node deployment where '2 of 3' nodes are required for each of these four component quorums. Generalization to N>1 is straightforward.

The *vRouter* is the forwarding element that acts as a data plane (DP) for the workloads running on the Compute servers across the cluster, replacing any native solution for workload networking. The vRouter is composed of two parts, the vRouter agent running in user space and the vRouter module running in kernel space (optionally replaced by the vRouter DPDK module running in user space). The vRouter agent performs all policy evaluation (such as security, NAT, mirroring, multicast, or load-balancing) by examining the flows going through the vRouter. The routing information is automatically imported and exported based on network policies defined on the Controller, and the vRouter performs all necessary bridging or routing. OpenContrail uses overlay networking to carry workload traffic between Compute nodes, in order to provide flexibility to the virtual network topology created on top of the fabric/underlay network. The vRouter is in charge of encapsulating traffic egressing the server.

Fig. 1 shows the totality of relevant OpenContrail 3.x processes within each node (role). In addition to the unique processes within each role described previously, each of the roles has a *supervisor* process and a *nodemgr* process. These common processes are described in more detail in section III.

III. Software Failure Mode Analysis

In the process failure mode description to follow, we assume that N=1 in the 2N+1 deployment configuration described above. That is, we consider a minimum 3-node deployment where '2 of 3' nodes are required for Controller quorum. However, at the process level not all processes must meet this '2 of 3' requirement. As described below, some processes do require '2 of 3' instances to be up, but some require only '1 of 3' instances, and some in fact are not strictly required at all ('0 of 3' instances).

Two processes are common across all OpenContrail roles, the *supervisor* and the *nodemgr* processes. Specifically, there are five *supervisors* and five *nodemgrs* common to the roles.

Each *supervisor* is dedicated to its respective node and role. From a resiliency standpoint, the main function of the *supervisor* is to auto-restart any failed processes within its node-role. Failure of the *supervisor* itself requires the entire node-role to be restarted (either immediately or at a later time such as a maintenance window). At the time of node-role restart, the other processes in the node-role must be manually killed, and the *supervisor* must be manually restarted, so that it can then auto-restart the other processes under its oversight. Until then, any process failures within that node-role require manual restart, but the remaining node-role functionality is unimpaired. In fact, the *supervisor* is a '0 of 3' process; all instances can fail and functionality is unimpaired.

Role	Process
Config	config-api
	discovery
	schema
	svc-monitor
	ifmap
	device-manager
	config-nodemgr
	supervisor-config
Control	control
	dns
	named
	control-nodemgr
	supervisor-control

Role	Process
Analytics	analytics-api
	alarm-gen
	collector
	query-engine
	redis
	analytics-nodemgr
	supervisor-analytics
Database	database (Config)
	zookeeper (Config)
	database (Analytics)
	kafka (Analytics)
	database-nodemgr
	supervisor-database

Role	Process
Vrouter	vrouter-agent
	vrouter-dpdk
	vrouter-nodemgr
	supervisor-vrouter

Figure 1. OperContrail 3.x processes by role.

Each *nodemgr* is dedicated to its respective node and role. If a *nodemgr* fails, other processes in that node-role continue to function, but process state visibility is lost (process status data is not fed to the associated Analytics *collector* process) until the *nodemgr* is auto-restarted by its *supervisor*, but the remaining node-role functionality is unimpaired. In fact, the *nodemgr* is also a '0 of 3' process; all instances within a role can fail and functionality is unimpaired.

With the exception of the *supervisor* processes, failure of any instance of any process of any role in any node does not impact the entire node-role in which it resides; that is, the other functioning processes in the node-role continue to function and are not impacted when that process is restarted.

Table I summarizes the node processes and failure modes. All Config, Control, and vRouter processes are auto-restarted by their *supervisor*. All Analytics processes are auto-restarted by their *supervisor* except for the *redis* process, which is not under *supervisor* control and requires manual restart. All Database processes require manual restart except for the *nodemgr* process, which is auto-restarted by the *supervisor*.

The *control* process plays a special role in host vRouter DP functionality. In particular, the per-host *vrouter-agent* process (described later) is normally connected in round-robin fashion to two *control* processes in separate Control nodes at any point in time. Thus, normally roughly equal numbers of all host *vrouter-agent* processes are connected to *control-1* and *-2*, to *control-1* and *-3*, and to *control-2* and *-3*.

If *control-1* fails, all *vrouter-agent* processes connected to *control-1* will rediscover (typically within a minute) the unused *control* process, and every *vrouter-agent* will then be connected to *control-2* and *control-3*. If *control-2* then fails, every *vrouter-agent* will then be connected to only *control-3*. In this scenario, the host DPs are not interrupted because every *vrouter-agent* is always connected to at least one *control* process. If *control-3* subsequently fails, then every host DP will go down because BGP forwarding tables will be flushed.

TABLE I. OPENCONTRAIL 3.X NODE PROCESS AND FAILURE MODES

Role	Process Name	SDN CP	Host DP
Config	*config-api*	1 of 3	0 of 3
	discovery	1 of 3	1 of 3
	schema	1 of 3	0 of 3
	svc-monitor	1 of 3	0 of 3
	ifmap	1 of 3	0 of 3
	device-manager	1 of 3	0 of 3
Control	*control*	1 of 3	1 of 3
	dns	0 of 3	1 of 3
	named	0 of 3	1 of 3
Analytics	*analytics-api*	1 of 3	0 of 3
	alarm-gen	1 of 3	0 of 3
	collector	1 of 3	0 of 3
	query-engine	1 of 3	0 of 3
	redis	1 of 3	0 of 3
Database	*cassandra-db* (Config)	2 of 3	0 of 3
	cassandra-db (Analytics)	2 of 3	0 of 3
	kafka	2 of 3	0 of 3
	zookeeper	2 of 3	0 of 3
vRouter	*vrouter-agent*	0 of 1	1 of 1
	vrouter-dpdk	0 of 1	1 of 1

In the unlikely event that two *control* processes fail simultaneously, then the one-third of *vrouter-agent* processes connected to those two Control nodes will drop packets until the affected *vrouter-agent* processes connect to the remaining *control* process. Once connection is reestablished, the host DP will resume without the need for process restart. For the analysis to follow, we assume that the impact of simultaneous *control* process failures on host DP availability is negligible.

Similar behavior applies to the *dns* and *named* processes. Each host is attached to two Control nodes, where DNS requests from VMs on that host are processed. The '1 of 3' requirement for host DP availability means that we must have at least one of the collection {*control* + *dns* + *named*} on the same Control node available. For example, having only *control-1* and *dns-2* and *named-3* available is not sufficient for host DP availability, and packets will be dropped. Thus, the collection {*control* + *dns* + *named*} is a '1 of 3' process block required for DP availability across all hosts.

The four Database processes are the only '2 of 3' quorum processes in the Controller, and a lack of quorum of any of these processes only impacts the SDN CP, not the host DP.

Any *vrouter-agent* or *vrouter-dpdk* process failure takes down the DP for the entire host. The agent is essential to evaluate what policy to apply to a flow, so if a *vrouter-agent* fails it will impact the DP traffic for that host. Prefixes of VMs sitting on that host will disappear from the routing tables/advertisements. In a DPDK deployment, the vRouter runs in user space and uses the DPDK library to optimize access to the underlying HW. If the *vrouter-dpdk* process fails, then the vRouter function cannot be executed.

IV. REFERENCE DEPLOYMENT TOPOLOGY

Theoretical availability analysis of any distributed SDN controller requires that we define a specific HW deployment topology. In particular, we must specify layout of Controller roles on VMs or containers, as well as layout of Controller node VMs on physical host servers and on common HW elements such as racks. Specifically, we consider three scenarios called Small, Medium, and Large. We again consider the minimal case of a 2N+1=3 Controller node cluster for the analysis. The approach can easily be extended to larger cluster configurations.

In the *Small* topology (top left of Fig. 2), the four critical Controller roles (confi<u>G</u>, <u>C</u>ontrol, <u>A</u>nalytics, <u>D</u>atabase) run within a single VM (labeled GCAD). Each of the three Controller node VMs (GCAD1-3) run on three separate hosts (H1-3) in a single shared rack (R1). Next, in the *Medium* topology (top right of Fig. 2), the four Controller roles run in separate VMs (G1-3, C1-3, A1-3, and D1-3). Each of the three sets of Controller node VMs run on three separate hosts (that is, G1 … D1 all run on H1, G2 … D2 on H2, and G3 … D3 on H3). Hosts H1 and H2 still reside in a shared rack (R1) but now H3 resides in a separate rack (R2). Finally, in the *Large* topology (bottom right of Fig. 2), the four Controller roles again run in separate VMs, but now each of the 12 Controller node VMs runs on its own separate host (G1 on H1, C1 on H2, …, D3 on H12). Each of the three sets Controller node VMs and hosts now run in three separate racks (H1-4 running node 1 VMs G1 … D1 in R1, H5-8 in R2, and H9-12 in R3).

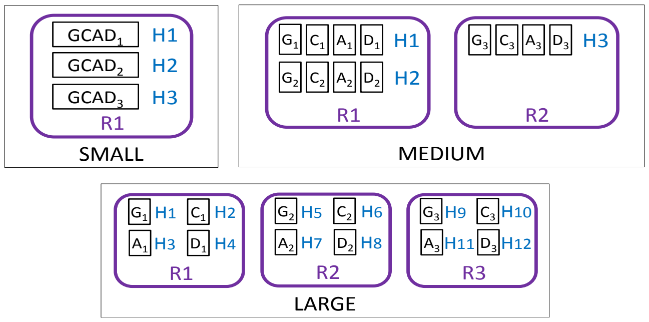

Figure 2. Reference hardware deployment topologies.

V. HW-Centric Availability Analysis

Given the three reference HW deployment topologies, we can compute the Controller availabilities. Let A_C denote the availability of an individual instance of any Controller role, and assume for simplicity that the availability of each role is the same (i.e., the availabilities of each instance of Config, Control, Analytics, and Database roles are all equal). Let A_V denote availability of an individual VM (including guest OS). Let A_H denote availability of a host (including host OS and hypervisor). Let A_R denote availability of a rack. Finally, Let $A_{m/n}(\alpha)$ denote availability of at least m of n parallel blocks, where α denotes the availability of a single block. Then

$$A_{m/n}(\alpha) = \begin{cases} \sum_{i=0}^{n-m} \binom{n}{i} \alpha^{n-i}(1-\alpha)^i & for\ m \le n \\ 0 & for\ m > n \end{cases}. \quad (1)$$

Block availability α can represent a single element, such as a role instance, or a combination of elements in series, such as {role+VM+host}. For simplicity, shorthand notation $A_{m/n}$ is used in lieu of $A_{m/n}(\alpha)$ in the results to follow, and the value of α is specified in the particular context.

For this HW-centric analysis, we do not consider individual processes within a role. Rather, we assume that each of the 12 Controller nodes is an atomic element. Thus, in a 3-node cluster, at least 1 out of 3 nodes of the Config, Control, and Analytics roles must be available for the Controller to be available, and at least 2 out of 3 nodes of the Database role must be available.

A. Small Topology

In the Small topology, each node's roles run on the same VM and host. In order to quantify availability, we must first condition on the combined {VM+host} availability, namely

$$A = \begin{bmatrix} (A|3\ hosts\ up)A_V{}^3 A_H{}^3 \\ +(A|2\ hosts\ up)3A_V{}^2 A_H{}^2(1-A_V A_H) \\ +(A|1\ host\ up)3A_V A_H(1-A_V A_H)^2 \end{bmatrix} A_R. \quad (2)$$

In the third term, $(A|1\ host\ up) = 0$ since the '2 of 3' quorum requirement for the Database role is violated if only one host is up. Next, in the second term, with two hosts up we need '1 of 2' nodes to be available for three roles (Config, Control, and Analytics) and '2 of 2' nodes to be available for the Database role. Thus $(A|2\ hosts\ up) = A_{1/2}{}^3 A_{2/2} = A_C{}^5(2-A_C)^3$. Finally, in the first term, with three hosts up we need '1 of 3' nodes to be available for three roles and '2 of 3' nodes for one role. Thus, $(A|3\ host\ up) = A_{1/3}{}^3 A_{2/3} = A_C{}^5\left(3-3A_C+A_C{}^2\right)^3(3-2A_C)$. In summary, the Small topology Controller availability A_S is given by

$$A_S = \begin{bmatrix} A_{1/3}{}^3 A_{2/3} A_V A_H \\ +3A_{1/2}{}^3 A_{2/2}(1-A_V A_H) \end{bmatrix} A_V{}^2 A_H{}^2 A_R \quad (3)$$

where $\alpha = A_C$ in (1) for expressions $A_{m/n}$. It can be shown that $A_S \approx A_{2/3} A_R$, where $\alpha = A_C A_V A_H$ in (1) for $A_{2/3}$. This simplified approximation makes intuitive sense; in the Small topology, {role+VM+host} is equivalent to a single element (of which 2 of 3 are required) in series with the single rack.

B. Medium Topology

In the Medium topology, the four Controller roles run in separate VMs and the hosts reside in two racks. In this case, the term α in (1) for the expressions $A_{m/n}$ represents the combined {role+VM} availability $\alpha = A_C A_V$. To quantify availability, we first condition on rack availability, namely

$$A = \begin{bmatrix} (A|R_1\ up\ R_2\ up)A_R{}^2 \\ +(A|R_1\ up\ R_2\ down)A_R(1-A_R) \\ +(A|R_1\ down\ R_2\ up)A_R(1-A_R) \end{bmatrix}. \quad (4)$$

In the third term, $(A|R_1\ down\ R_2\ up) = 0$ since the Database quorum requirement is violated if rack 1 is down. In the second term, $(A|R_1\ up\ R_2\ down) = A_{1/2}{}^3 A_{2/2} A_H{}^2 = \alpha^5 A_H{}^2(2-\alpha)^3$. Finally, the first term requires that we further condition on the host availability, namely

$$\begin{aligned}(A|R_1\ up\ R_2\ up) = (A|3\ hosts\ up)A_H{}^3 \\ +(A|2\ hosts\ up)3A_H{}^2(1-A_H) \\ +(A|1\ host\ up)3A_H(1-A_H)^2.\end{aligned} \quad (5)$$

$(A|1\ host\ up) = 0$ since the Database node quorum is violated if only one host is up. Next, $(A|2\ hosts\ up) = A_{1/2}{}^3 A_{2/2} = \alpha^5(2-\alpha)^3$. The first term $(A|3\ hosts\ up) = A_{1/3}{}^3 A_{2/3} = \alpha^5(3-3\alpha+\alpha^2)^3(3-2\alpha)$. Thus, Medium topology Controller availability A_M is given by

$$A_M = \begin{bmatrix} A_{1/3}{}^3 A_{2/3} A_H \\ +A_{1/2}{}^3 A_{2/2}(4-3A_H-A_R) \end{bmatrix} A_H{}^2 A_R \quad (6)$$

where $\alpha = A_C A_V$ in (1) for $A_{m/n}$. Again, it can be shown that $A_M \approx A_{2/3} A_R \approx A_S$, where $\alpha = A_C A_V A_H$ in (1) for $A_{2/3}$. This approximation also makes intuitive sense; in Medium topology, {DB role + DB VM + host} is equivalent to a single element (of which 2 of 3 are required) in series with the rack hosting the quorum of elements. The other 1 of 3 {role+VM} elements have only second-order effects on availability.

C. Large Topology

Finally, in the Large topology, each of the 12 Controller VMs runs on its own host and each Controller node resides in its own rack. In this case, the term α in (1) for the expressions $A_{m/n}$ represents the combined {role+VM+host} availability $\alpha = A_C A_V A_H$. In order to quantify availability, we must again first condition on the rack availability, namely

$$A = \begin{bmatrix} (A|3\,racks\ up){A_R}^3 \\ +(A|2\,racks\ up)3{A_R}^2(1-A_R) \\ +(A|1\,rack\ up)3A_R(1-A_R)^2 \end{bmatrix}. \qquad (7)$$

$(A|1\,rack\ up) = 0$ since the Database node quorum is violated if only one rack is up. Next, $(A|2\,racks\ up) = {A_{1/2}}^3 A_{2/2} = \alpha^5(2-\alpha)^3$. The first term $(A|3\,racks\ up) = {A_{1/3}}^3 A_{2/3} = \alpha^5(3-3\alpha+\alpha^2)^3(3-2\alpha)$. Thus, the Large topology Controller availability A_L is given by

$$A_L = \left[{A_{1/3}}^3 A_{2/3} A_R + {A_{1/2}}^3 A_{2/2} 3(1-A_R)\right]{A_R}^2 \qquad (8)$$

where $\alpha = A_C A_V A_H$ in (1) for $A_{m/n}$. Now, it can be shown that $A_L \approx A_{2/3}$, where $\alpha = A_C A_V A_H A_R$ in (1) for $A_{2/3}$. This approximation also makes intuitive sense; in Large topology, {DB role + DB VM + DB host + rack} is effectively a single element (of which 2 of 3 are required). The other 1 of 3 {role+VM+host} elements have only second-order effects.

D. Comparative HW-Centric Availability Results

Based on rules of thumb from years of Telecom industry experience, we assume in the example to follow that $A_C = 0.9995$, $A_V = 0.99995$, $A_H = 0.9999$, and $A_R = 0.99999$. These values are intended to represent ballpark parameters, and the results to follow are intended for relative, qualitative comparisons (rather than absolute, quantitative assessments). The resulting relative comparisons and observations remain the same regardless of the actual values used. Large-scale studies of enterprise failure rates do exist (c.f. [14]), but industry-standard failure rates are difficult to establish [15]. And even if statistically reliable failure data were readily available, availabilities will depend on recovery procedures.

Actual parameter values depend heavily on the particular controller SW fault rates and auto-recovery capabilities, VM and container technologies, server HW failure rates, vendor maintenance contracts, etc. Thus, SW and HW availabilities will always depend on the chosen technology. For example, enterprise-grade servers may have a mean-time-between-failure (MTBF) in the 5-year range [16]. In a hardened Telco data center with spare HW on-site and 24x7 staffing, the maintenance level is considered Same Day (SD), typically corresponding to a 4-hour mean-time-to-restore (MTTR). In a cloud data center, the maintenance contract could be Next Day (ND), typically translating to a 24-hour MTTR after accounting for intra-day incident timing, or Next Business Day (NBD), typically translating to a 48-hour MTTR after also accounting for intra-week incident timing [17]. Given these various scenarios, A_H = MTTF/(MTTF+MTTR) can range from 0.9990 (NBD) to 0.9995 (ND) to 0.9999 (SD).

Fig. 3 shows OpenContrail Controller availability as a function of Controller role availability $A_C \in [0.9995 \pm 0.0005]$ for the Small, Medium, and Large reference HW topologies, with $A_V = 0.99995$, $A_H = 0.99990$, and $A_R = 0.99999$. As can be seen, with role availability A_C = 0.9995, Controller availability is 0.999989 for the Small and Medium topologies and 0.999999 for the Large topology. As the role availability A_C ranges between 0.999 and 1.0, the Small and Medium availabilities range between 0.999986 and 0.999990 while Large availability ranges between 0.999996 and 0.9999999.

We can draw several important conclusions from this HW-centric analysis. First, although separation of roles onto separate VMs (S→M) offers the ability to independently scale roles for capacity planning, it does not improve availability. With role separation, there are 4x as many VMs to fail, but the impact of each VM failure is roughly ¼. The same is true for host separation (M→L). Although this effect is masked by the impact of the additional rack separation (discussed later), separation of VMs onto separate hosts does not improve availability. With VM separation, there are 4x as many hosts to fail but the impact of each failure is roughly ¼.

Second, rack separation does not improve availability unless three racks are employed. In fact, contrary to expectation, adding a second rack (S→M) actually slightly reduces availability, since the '2 out of 3' quorum still exists on a single rack. In both the Small and Medium topologies, failure of rack R1 causes complete Controller failure because two of the three nodes reside on that one rack. But in the Medium topology, we have added a second rack R2 and its associated failure modes. So there are 2x as many racks to fail, but the impact of a rack failure is > ½ because of the overweight significance of rack R1 failures.

In contrast, adding the third rack (M→L) does improve availability. With A_C = 0.9995, Controller availability increases from 0.999989 to 0.999999 (a savings of 5 minutes/year in downtime), since the '2 of 3' quorum is not broken by a single rack failure. From an availability standpoint, the conclusion is clear: *one rack or three, but not two*. Furthermore, the space and expense of multiple racks must be weighed against the relatively modest improvement in availability between the Small and Large deployments.

In conclusion, this high-level approach offers a simple yet powerful methodology to quantify Controller availability as a function of key parameters. Perhaps more importantly, this HW-centric approach provides a means to quickly and easily

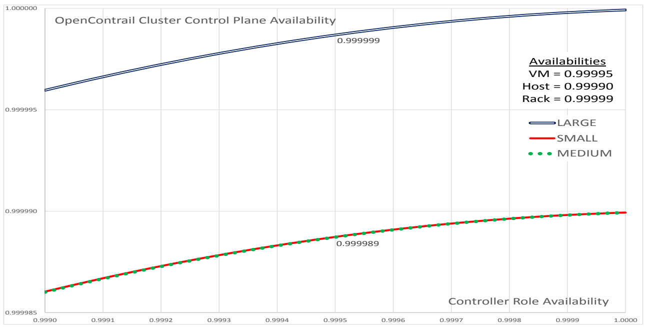

Figure 3. OpenContrail cluster availability (HW-centric).

perform relative sensitivity analyses on various possible HW deployment topologies, thus facilitating evaluation of the cost:resiliency tradeoff before capital investment occurs.

VI. SW-Centric Availability Analysis

As useful as we have shown it to be, the previous HW-centric approach still only treats the Controller roles as equal, atomic SW elements. In reality, the four role types each consist of distinct SW processes with distinct failure modes (outlined in section III). Thus, the assumptions in section V regarding equal role availability A_C for all roles and role-level quorum requirements can be further fine-tuned with a more SW-centric focus. Furthermore, this more detailed approach allows us to separately quantify availability of the SDN control plane (CP) and per-host vRouter data plane (DP). Based on the results in section V (in particular, the observations regarding the lack of resiliency impact of VM and host separation, and rack separation onto less than three racks), the SW-centric analysis will focus only on the two extreme cases of the Small (S) and Large (L) HW topologies.

A. Supervision and Restart

From a reliability modeling standpoint, the *supervisor* processes require special consideration. As discussed in section III, each critical Controller role (Config, Control, Analytics, and Database) has its own *supervisor* process. Thus, there are 3 nodes x 4 roles = 12 *supervisor* processes in a 3-node configuration, plus the per-host *supervisors* for the vRouter DP role. The *supervisor* is responsible for auto-restarting failed processes within their respective node-role.

If a particular *supervisor* process dies, the other *supervisors* in any other nodes or roles do not assume responsibility for that node-role. For example, if the *supervisor* in Config node 2 dies (call this process *supervisor-config-2*), then every other process in the *config-2* node is running in an unsupervised mode. Neither the *supervisors* for other Config nodes, nor the *supervisors* for other node 2 roles, can assume supervision of Config node 2. All *config-2* processes remain unsupervised until *supervisor* restart.

Furthermore, as described in section 3.1, if a particular *supervisor* dies, any failed processes within that *supervisor's* node-role must be manually restarted until the *supervisor* itself is manually restarted, and manual restart of a *supervisor* requires that every process in its node-role be killed prior to *supervisor* restart so that the *supervisor* can then auto-restart those processes under its oversight and control.

These behaviors lead to the need to analyze two cases:

- The optimistic upper bound case where the *supervisor* is not required for continued node-role operation. In this option, we assume that the node-role will be brought down and the *supervisor* restarted during the next maintenance window in a hitless manner.
- The realistic lower bound case where the *supervisor* is required for continued operation. In this option, we assume that all node-role processes are killed and the *supervisor* restarted immediately following its failure.

These options impact the process availability parameters used in the analysis to follow. Let A denote availability of an individual process under *supervisor* control (auto-restarted), and let A_S denote availability of an individual process that is unsupervised and requires manual restart (as an example, the *supervisor* process itself). Let F denote the process MTBF, let R denote the mean time to auto-restart a *supervisor*-controlled process, and let R_S denote the mean time to manually restart an unsupervised process such as the *supervisor*. Then $A = F/(F{+}R)$ and $A_S = F/(F{+}R_S)$.

Assume *for simplicity* that F, R, and R_S are the same for all associated processes in all roles (thus A and A_S are the same for all supervised and unsupervised processes). As we will show, it is straightforward to relax this assumption in the following methodology, if appropriate. We have created two types of processes already, and we can easily expand to K process types if lab/field data for F suggest the need to do so.

We assume in the example results to follow that $A = 0.99998$ (based on $F = 5000$ hours and $R = 0.1$ hour) and $A_S = 0.99980$ (based on $R_S = 1$ hour), along with the previous values for A_V, A_H, and A_R. These variables are parameters in the model and can easily be changed. Again, the results to follow are intended for relative, qualitative comparisons (rather than absolute assessments). In order to reflect differing degrees of SW process maturity and auto-recovery capabilities with different SDN controller implementations, sensitivity analysis is performed as a function of process availability $A \in [0.99998 \pm 1$ order of magnitude].

If the *supervisor* is not required, then the actual process restart time depends on when the process fails during the (say 10 hour) interval after the *supervisor* fails but before the next opportunity to schedule a maintenance window. Assuming F ~ exponential, then Pr{failure during 10h *supervisor* outage} $= 1{-}e^{-10/F} = 0.002$ and the actual process restart time $R^* = (e^{-10/F})R{+}(1{-}e^{-10/F})R_S = 0.102$ hours. Thus, the actual process availability $A^* = F/(F{+}R^*) \approx 0.99998$. That is, process availability A is not measurably impacted in this scenario 1.

In contrast, if the *supervisor* is required, then either the failure of a process or that of its *supervisor* causes the process to restart. In this case, the actual process failure interval $F^* = F/2 = 2500$ hours (assuming equal failures rates) and the actual restart time $R^* = (R_S{+}R)/2 = 0.55$ hours. Thus, the actual process availability $A^* = F^*/(F^*{+}R^*) \approx 0.9998$. That is, every process effectively inherits the *supervisor* availability A_S in this scenario 2.

As mentioned, some processes are not under *supervisor* control and thus require manual restart (e.g., *redis*). Table II lists the counts of processes within each role requiring auto and manual restart. When the *supervisor* is not required, processes in the four Auto columns have availability A and those in the four Manual columns have availability A_S. When the *supervisor* is required, all processes effectively have availability A_S regardless of their default restart mode. If additional process types are needed to account for different F (e.g., new vs. mature code), these counts in Table II can be further broken down (e.g., New-Auto, Mature-Auto, etc.).

B. Quorum Requirements

As discussed in section III, each Controller process has different quorum requirements for CP and DP availability. A few processes (notably within the Database role) require '2 of 3' instances to be available, most require only '1 of 3' instances to be available, and some are not strictly required at all for CP or DP functionality. Let M_R denote the number of role R processes requiring '2 of 3' instances available, and let N_R denote the number of role R processes requiring '1 of 3' instances available, where R = G (Config), C (Control), A (Analytics), or D (Database). Table III shows the values for M_R and N_R based on the failure mode analysis in section III. The analyses to follow are expressed in terms of variables M_R and N_R for generality and extensibility to other controllers.

Tables II and III fully encapsulate OpenContrail 3.x. Other Controller implementations can be accommodated simply by modifying the rows, columns, and values in these tables. For example, we model two process types (from an availability standpoint): Auto and Manual. If a different SDN controller has more than two distinct recovery modes, or processes with distinct maturity levels, then additional rows can be added to Table II. We model four nodes types. If a different controller has functionality partitioned differently, then columns (rows) can be added to or removed from Table II (III). If a different controller has a different process count required for CP and DP operability, then values in each cell of Tables II and III can be changed. Although the analysis in the next sections may rightfully seem tedious, inclusion of this analysis is essential to demonstrate how these differences among different SDN controllers easily translate into simple changes in the mathematical formulation.

C. *Small Topology, Supervisor Not Required (1S)*

We first consider the (optimistic upper bound) case where the *supervisor* processes are not required in the Small topology (option 1S). As in section V, in order to quantify availability, we must first condition on the {VM+host} availability. In this case, the Small topology SDN CP availability A_{CP} for the 1S option is given by

$$A_{CP} = \begin{bmatrix} (A|3\ hosts\ up)A_V{}^3A_H{}^3 + \\ (A|2\ hosts\ up)3A_V{}^2A_H{}^2(1-A_VA_H) \end{bmatrix} A_R, \quad (9)$$

$$\text{where } (A|x\ hosts\ up) = (A_{2/x})^{\sum \mathrm{M_R}}(A_{1/x})^{\sum \mathrm{N_R}}, \quad (10)$$

TABLE II. COUNTS OF PROCESSES BY RESTART MODE BY ROLE

Restart Mode	Config	Control	Analytics	Database
Auto	6	3	4	0
Manual	0	0	1	4

TABLE III. COUNTS OF PROCESSES BY QUORUM TYPE BY ROLE

Role	SDN CP		Host DP	
Quorum Type	***M***	***N***	***M***	***N***
Config G	0	6	0	1
Control C	0	1	0	1*
Analytics A	0	5	0	0
Database D	4	0	0	0
Sums	4	12	0	2

* {control+dns+named} is a '1 of 3' process block required for DP availability, modeled as single process with availability A^3.

and $\alpha = A$ in (1) for the expressions $A_{m/n}$ and $\sum M_R$ and $\sum N_R$ denote the sums for all roles $R = G, C, A, D$ for the SDN CP in Table III. Thus, the Small topology SDN CP availability A_{CP} for the 1S option is given by

$$A_{CP} = \Big[(A_{2/3})^{\sum \mathrm{M_R}}(A_{1/3})^{\sum \mathrm{N_R}}A_V{}^3A_H{}^3 \\ +(A_{2/2})^{\sum \mathrm{M_R}}(A_{1/2})^{\sum \mathrm{N_R}}3(1-A_VA_H)A_V{}^2A_H{}^2\Big]A_R . \quad (11)$$

Intuitively, this expression makes sense. CP availability is the sum of the conditional probabilities of functional availability in two cases, when 3 of 3 {VM+host} blocks are up and when 2 of 3 blocks are up, each weighted by their corresponding probabilities. Functional availability is given by the product of individual role availabilities, consisting of $\sum M_R$ quorum processes and $\sum N_R$ non-quorum processes. This sum is multiplied by the availability of the shared rack.

The host DP availability consists of two components, the *shared* contribution from Controller-based roles (impacting the DP for every host) and the *local* contribution from the host-based vRouter role (impacting the DP only for that host). The expression for *shared* DP availability A_{SDP} is identical to (11) except that exponents $\sum M_R$ and $\sum N_R$ are instead based on the Host DP columns in Table III. Next, the expression for the *local* DP contribution to availability $A_{LDP} = A^K$, where (again) A denotes individual process availability, and K denotes the number of host-based vRouter processes that must be available. (K = 2 in OpenContrail, *vrouter-agent* and *vrouter-dpdk*.) Combining, the Small topology per-host DP availability A_{DP} is given by $A_{DP} = A_{SDP}\,A_{LDP} = (A_{SDP})A^K$.

D. *Small Topology, Supervisor Required (2S)*

We next consider the (realistic lower bound) case where the *supervisor* processes are required in the Small topology (option 2S). Again conditioning on {VM+host} availability, the Small topology SDN CP availability A_{CP} for the 2S option is given by (9). In order to quantify availability in the case when the *supervisor* processes are required, we must further condition on the *supervisor* process availability, namely

$$(A|x\ hosts\ up) = \\ \sum_{g=1}^{x}\sum_{c=1}^{x}\sum_{a=1}^{x}\sum_{d=1}^{x}(A|g,c,a,d)P(g,c,a,d|x). \quad (12)$$

$(A|g,c,a,d)$ denotes the Controller availability with g Config, c Control, a Analytics, and d Database *supervisor* process instances available, given by

$$(A|g,c,a,d) = (A_{2/g})^{\mathrm{M_G}}(A_{1/g})^{\mathrm{N_G}}(A_{2/c})^{\mathrm{M_C}}(A_{1/c})^{\mathrm{N_C}} \\ \times (A_{2/a})^{\mathrm{M_A}}(A_{1/a})^{\mathrm{N_A}}(A_{2/d})^{\mathrm{M_D}}(A_{1/d})^{\mathrm{N_D}}, \quad (13)$$

where $\alpha = A$ in (1) for $A_{m/n}$, and M_R and N_R denote the role R values for the SDN CP in Table III.

Next, $P(g,c,a,d|x)$ denotes the probability (given x hosts up) of g Config, c Control, a Analytics, and d Database *supervisor* process instances available, given by

$$P(g,c,a,d|x) = \binom{x}{g}\binom{x}{c}\binom{x}{a}\binom{x}{d} \left[\rho^{g+c+a+d}(1-\rho)^{4x-g-c-a-d}\right], \quad (14)$$

where $\rho = A_S$ is the *supervisor* process instance availability.

Combining, the SDN CP availability A_{CP} for the 2S option is given by (9,12-14) where $\alpha = A$ in (1) for $A_{m/n}$ in (13), M_R and N_R in (13) denote the role R values for the SDN CP in Table III, and $\rho = A_S$ in (14). Again, the expression for the *shared* DP availability A_{SDP} is identical to (9,12-14) except that exponents M_R and N_R in (13) are instead based on the Host DP rows in Table III.

Again, this expression for CP availability (while dense) makes intuitive sense. CP availability is still the sum of the weighted conditional functional availabilities in two cases, 3 of 3 {VM+host} blocks up and 2 of 3 blocks up, but when *supervisor* processes are required, conditional availabilities are more complicated. Now, functional availability must be further conditioned on *supervisor* process availability. There are 4x3 possible cases with three {VM+host} blocks up, and 4x2 cases with two blocks up, given in (12). For instance, $(A|3,1,2,3)$ denotes the conditional availability with 3 Config, 1 Control, 2 Analytics and 3 Database *supervisor* instances up. The corresponding probability weights are more complicated as well. There are multiple combinations with the appropriate numbers of *supervisor* instances up or down, as given in (14). For instance, $P(3,1,2,3|3)$ denotes the conditional probability of 3 Config, 1 Control, 2 Analytics, and 3 Database *supervisor* process instances up and the rest down, given 3 {VM+host} blocks up.

Finally, the expression for the *local* DP contribution to per-host DP availability when the host vRouter *supervisor* process is required $A_{LDP} = A^K A_S$, where (again) A is the individual process availability, K denotes the number of host-based vRouter processes that must be available, and A_S is the individual *supervisor* process availability. Combining, the per-host DP availability A_{DP} is given by $A_{DP} = A_{SDP}\, A^K A_S$.

E. Large Topology, Supervisor Not Required (1L)

We next consider the (optimistic upper bound) case where *supervisor* processes are not required in the Large topology (option 1L). Again, in order to quantify availability, we must first condition on the rack availability. In this case, the Large topology SDN CP availability A_{CP} for 1L option is given by

$$A_{CP} = (A|3\ racks\ up){A_R}^3 + (A|2\ racks\ up)3{A_R}^2(1-A_R). \quad (15)$$

$(A|x\ racks\ up)$ is given by (12-14) where x now denotes the number of available racks rather than hosts; g, c, a, and d denote the number of available Config, Control, Analytics, and Database {VM+host} blocks rather than *supervisor* process instances; and $\rho = A_V A_H$. Combining, the SDN CP availability A_{CP} for 1L option is given by (15,12-14) where $\alpha = A$ in (1) for $A_{m/n}$ in (13), M_R and N_R in (13) denote role R values for the SDN CP in Table III, and $\rho = A_V A_H$ in (14). Intuition for this expression mimics that of option 1S.

Again, the *shared* DP availability A_{SDP} is identical to (15,12-14) except that exponents M_R and N_R in (13) are instead based on the Host DP rows in Table III. Next, the *local* DP availability $A_{LDP} = A^K$. Combining, the per-host DP availability A_{DP} is given by $A_{DP} = A_{SDP}\, A^K$.

F. Large Topology, Supervisor Required (2L)

Finally, we consider the (realistic lower bound) case where the *supervisor* processes are required in the Large topology (option 2L). As before, we must first condition on the rack availability. We then further condition on the {*supervisor*+VM+host} availability. The analysis is identical to that of option 1L except that $\rho = A_S A_V A_H$ in (14) rather than $A_V A_H$. That is, the Large topology SDN CP availability A_{CP} for 2L option is given by (15,12-14) where $\alpha = A$ in (1) for $A_{m/n}$ in (13), M_R and N_R in (13) denote the role R values for the SDN CP in Table III, and $\rho = A_S A_V A_H$ in (14). Intuition for this expression mimics that of option 2S.

Again, the *shared* DP availability A_{SDP} is identical to (15,12-14) except that exponents M_R and N_R in (13) are instead based on the Host DP rows in Table III, and *local* DP availability $A_{LDP} = A^K A_S$. Combining, the per-host DP availability A_{DP} is given by $A_{DP} = A_{SDP}\, A^K A_S$.

G. Comparative SW-Centric Availability Results

We assume in the example to follow that the (default) individual process availability A = 0.99998 and the (default) supervisor process availability A_S = 0.99980, along with the previous values for A_V, A_H, and A_R. The figures show OpenContrail availability as a function of process availability $A \in$ [0.99998 ± 1 order of magnitude of downtime (DT)]. That is, the center of the x-axis range (value of 0) corresponds to defaults A = 0.99998 and A_S = 0.9998. The left-most side of the x-axis (with value of –1) corresponds to A = 0.9998 and A_S = 0.998 (1 order of magnitude less reliable, i.e., 10x more DT), and the right-most side of the x-axis (value of 1) corresponds to A = 0.999998 and A_S = 0.99998 (1 order of magnitude less DT). A and A_S are varied in lock-step.

First, Fig. 4 shows SDN CP availability A_{CP} as a function of A for the Small and Large HW deployment topologies with and without *supervisor* processes required. As can be seen, with default individual process availability A = 0.99998, A_{CP} exceeds 0.999987 for the Small topology and 0.999997 for the Large topology. Requiring the *supervisor* increases downtime from 5.9 to 6.6 minutes/year (m/y) in the Small topology and from 0.7 to 1.4 m/y in the Large topology.

The addition of two racks to create the Large topology saves 5 m/y of CP DT. While this savings sounds modest relative to the cost of two additional racks, it is important to realize that this value is an average. In reality, the single-rack Small topology may experience no rack-related downtime for many years followed by a highly-publicized extended outage. A_R = 0.99999 could consist of a rack failure every 500 years, lasting two days to deliver new HW and rerack servers. But for a network or content or video service provider with 500

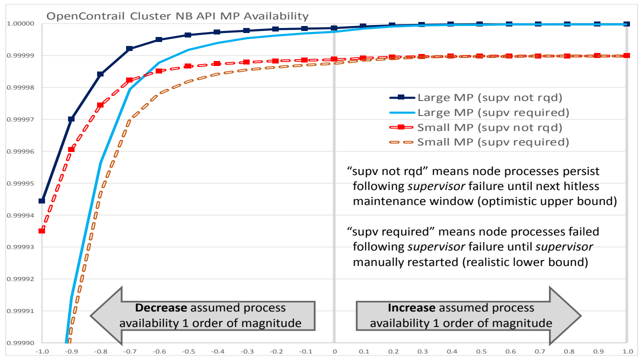

Figure 4. OpenContrail SDN CP Availability A_{CP} (SW-Centric).

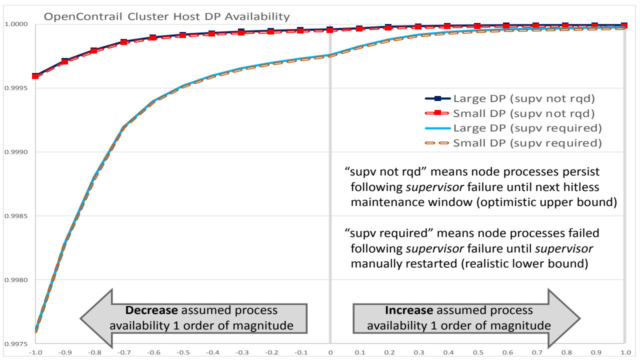

Figure 5. OpenContrail DP Availability A_{DP} (SW-Centric).

edge sites, a yearly outage may be unacceptable. The analysis shows that these outages can be avoided with rack separation.

If process availability A decreases by 1 order of magnitude to 0.9998 (and A_S to 0.998), CP availability A_{CP} decreases rapidly, the impact of rack separation becomes less relevant (Small and Large topologies begin to converge), and impact of the *supervisor* process becomes more pronounced. When *supervisor* is required, the dominant failure mode is: one Database *supervisor* failure and any Database process failure in another node, taking down two Database instances, resulting in quorum loss. When *supervisor* is not required, the dominant failure mode is: two failures of the same Database process in different nodes. The dominant SW failure mode in any implementation derives from the weakest link, typically the combination of quorum needs and process availability. In the case of OpenContrail, these factors converge for Database processes (all required for a quorum and all requiring manual restart), and the Database *supervisor* process in particular.

At the other extreme, when process availability A increases by 1 order of magnitude to 0.999998 and A_S to 0.99998, A_{CP} levels off quickly, the impact of the *supervisor* process becomes irrelevant, and the impact of rack separation in the Large topology becomes the key differentiator. The CP availabilities with and without the *supervisor* required converge to 0.99999 (Small topology) and to 0.9999998 (Large topology). The difference is due to rack separation.

Next, Fig. 5 shows host DP availability A_{DP} as a function of A for the Small and Large topologies with and without *supervisor* processes required. As can be seen, with process availability A = 0.99998, DP availability A_{DP} = 0.99975+ for both Small and Large topologies when vRouter *supervisor* is required, and 0.99995+ when the vRouter *supervisor* is not required. The difference is due entirely to the vRouter *supervisor*. Requiring the *supervisor* increases downtime by 5x from 26 to 131 m/y in the Small topology and by 6x from 21 to 126 m/y in the Large topology. Again, the third rack in the Large topology saves 5 m/y of SDP downtime. Other than this 5 m/y, there is little difference between the Small and Large topologies (as expected), since total DP availability is dominated by the identical host vRouter LDP availability.

If individual process availability A decreases 1 order of magnitude to 0.9998 (and A_S to 0.998), DP availability A_{DP} decreases rapidly, and *supervisor* process impact becomes more pronounced. Small and Large availabilities converge to 0.9976 (*supervisor* required) or to 0.9996 (*supervisor* not required). When the *supervisor* process is required, the dominant failure mode is failure of any *supervisor*. When the *supervisor* process is not required, the dominant failure mode is failure of either vRouter process. At the other extreme, when process availability A increases by 1 order of magnitude to 0.999998 (and A_S to 0.99998), DP availability A_{DP} levels off quickly. Small and Large DP availabilities converge to 0.999976 (*supervisor* required) or to 0.999996 (*supervisor* not required). Again, the difference is due to rack separation in the SDP contribution to total DP availability.

VII. Conclusions

Given the critical role that distributed Software Defined Networking controllers such as OpenContrail play in cloud computing and networking architectures, understanding their resiliency profile is crucial. In this work, we decompose the SDN controller node into its various roles, and drill down to the process level. We analyze the Controller process failure modes and their effects on the intra-element SDN control plane and the subtending host vRouter data planes within a cloud environment (data center). We develop HW- and SW-centric parametric availability models for a variety of HW topologies and SW modes of operation to predict availability and quantify sensitivity to platform and process resiliency.

Overall, the results suggest that the Controller SW can achieve very high levels of availability. In contrast, analysis of vRouter SW resiliency uncovered process single points of failure, and suggests that the host vRouter DP may achieve much lower availability and may limit host-level resiliency. Identifying these process weak links allows service provider operations to develop automation to reduce downtime and improve vRouter availability, and provides the Open Source community with focus areas for code improvements.

The high-level, HW-centric approach offers a simple yet powerful methodology to quantify Controller availability as a function of key parameters. More importantly, this approach provides a means to quickly perform relative sensitivity analyses on various HW deployment topologies, thus facilitating evaluation of the cost:resiliency tradeoff before capital investment occurs. For a HW deployment on one or two racks and a '2 of 3' quorum SW implementation, we show that the Controller availability is approximately given by $A \approx \alpha^2(3-2\alpha)A_R$, where $\alpha = A_C A_V A_H$, and A_C,

A_V , A_H, and A_R respectively denote the Controller node, VM, host, and rack availabilities. For a HW deployment on three racks, we show that Controller availability is approximately given by $A \approx \alpha^2(3-2\alpha)$, where instead $\alpha = A_C A_V A_H A_R$.

A number of important observations can be deduced regarding the underlying HW deployment topology. First, the separation of roles onto separate VMs does not improve availability. With role separation, there are 4x as many VMs to fail, but the impact of each VM failure is roughly ¼. The same is true for separation of the VMs onto separate hosts.

Second, rack separation does not improve availability unless three racks are employed. Contrary to expectation, expanding from one to two racks actually reduces availability. In both the one- and two-rack deployments, failure of a particular rack causes complete Controller failure because the '2 of 3' quorum still resides on one rack. But adding a second rack adds its associated failure modes. So there are 2x as many racks to fail, but the impact of a rack failure is > ½ because of the overweight significance of failures of the quorum rack. In contrast, adding a third rack does improve availability, since the quorum is not broken by the failure of any single rack. From an availability standpoint in a '2 of 3' quorum deployment, the conclusion is clear: one rack or three racks, but not two.

The space and expense of a second and third rack must be weighed against the availability improvement. The addition of two more racks saves a few minutes/year of downtime. While this sounds modest relative to the cost associated with two additional racks, it is important to realize that this is an average. In reality, the single-rack topology in each site will experience no rack downtime for many years followed by a highly-publicized extended outage. For a network or content service provider with many cloud sites, the resulting frequent high-profile outages may be unacceptable.

The detailed SW-centric approach developed in this paper yields additional insights that can only be gained from a process-level analysis. For instance, as individual process availability decreases, SDN CP availability decreases rapidly, the impact of rack separation becomes less relevant, and the impact of the *supervisor* process becomes more pronounced. At the other extreme, as individual process availability increases, CP availability quickly levels off, the impact of the *supervisor* process becomes irrelevant, and the impact of rack separation becomes the key differentiator. Next, as individual process availability decreases, host DP availability decreases rapidly, and the impact of the *supervisor* process becomes more pronounced. At the other extreme, as process availability increases, DP availability quickly levels off, and the impact of the *supervisor* process becomes less relevant. In all cases, the impact of rack separation remains constant. Again, the third rack saves a few minutes/year of downtime on average.

Although OpenContrail is used as an example, we develop a flexible, extensible analytic modeling framework that can be used to assess the relative availability of any distributed SDN controller. We encapsulate the Controller SW implementation into two tables and develop the mathematical framework around these inputs so that other implementations can be analyzed simply by populating the tables appropriately. The parametric model framework is thus highly flexible and easily adaptable to other distributed SDN controller implementations through straightforward changes to a well-defined set of parameters. Furthermore, we evaluate HW topologies spanning the extremes from the smallest reasonable scenario to the largest scenario necessary for complete quorum process separation.

To the best of our knowledge, this work represents the first effort of its kind to address the process-level dynamics and quantitatively assess intra-element SDN controller SW failure modes and availability at the Controller process level. Future work includes simulating the topologies to validate the conclusions, and leveraging the results to pinpoint areas for machine learning automation to reduce downtime and identify opportunities for Controller SW improvements.

Acknowledgment

We wish to thank our management for their continual support of this and other Open Source contributions.

References

[1] A. Singla and B. Rijsman, *Day One: Understanding OpenContrail Architecture*. Juniper Networks, Inc., 2013.

[2] https://docs.opendaylight.org/en/stable-fluorine/

[3] https://wiki.onosproject.org/display/ONOS/

[4] http://www.projectfloodlight.org/floodlight/

[5] https://www.vmware.com/products/nsx/solution-brief.pdf

[6] http://docs.openvswitch.org/en/latest/intro/what-is-ovs/

[7] F. Longo, S. Distefano, D. Bruneo, and M. Scarpa, "Dependability modeling of software defined networking," *Computer Networks*, vol. 83, pp. 280-296, 2015.

[8] T. Nguyen, T. Eom, S. An, J. Park, J. Hong, and D. Kim, "Availability modeling and analysis for software defined networks," *21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, pp. 159-168, 2015.

[9] G. Nencioni, B. Helvik, A. Gonzalez, P. Heegaard, and A. Kamisinski, "Availability modelling of software-defined backbone networks," *46th IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, pp. 105-112, 2016.

[10] K. Han, T. Nguyen, D. Min, and E. Choi, "An evaluation of availability, reliability and power consumption for a SDN infrastructure using stochastic reward net," *Advances in Computer Science and Ubiquitous Computing*, J. Park, Y.Pan, G. Yi, and V. Loia (editors). Singapore: Springer, 2016.

[11] P. Heegaard, B. Helvik, G. Nencioni, and J. Wafler, "Managed dependability in interacting systems," *Principles of Performance and Reliability Modeling and Evaluation*, L. Fiondella and A. Puliafito (editors). Switzerland: Springer, 2016.

[12] P. Vizarreta, P. Heegaard, B. Helvik, W. Kellerer, and C. Mas Machhura, "Characterization of failure dynamics in SDN controllers," *9th International Workshop on Resilient Networks Design and Modeling (RNDM)*, pp. 1-7, 2017.

[13] G. Nencioni, B. Helvik, and P. Heegaard, "Including failure correlation in availability modeling of a software-defined backbone network," *IEEE Trans. on Network and Service Management*, vol. 14 no. 4, 2017.

[14] B. Schroeder and G. Gibson, "A large-scale study of failures in high-performance computing systems," *International Conference on Dependable Systems and Networks (DSN2006)*, June, 2006.

[15] S.Miller, "Explaining the lack of large scale studies in IT," *SMB IT Journal*, March, 2015.

[16] Vendor-proprietary private communications under NDA, 2016.

[17] Y. Kogan, AT&T Labs internal memorandum, unpublished.