Dominant Network Slices

1st Bradley R. Smith
Curated Networks, Inc.
Santa Cruz, CA USA
brad@curatednetworks.com

***Abstract*—Network slicing is a key component of the 5G network architecture that enables support for the diverse needs modern network applications have of network services. The current state-of-the-art solution for implementing network slices, Segment Routing, has shortcomings including use of a limited abstraction ("colors") for Traffic Engineering (TE) requirements, and the use of least cost path(s) for each color. We propose a new approach that uses a Boolean algebra and partially ordered metrics to pre-compute a set of paths that satisfies the full range of performance and TE policies available in the network. This approach, in effect, computes a dominant set of paths (those where no other path has better performance) for each truth assignment to the Boolean expressions. New flows are routed over the least-congested path that satisfies the flow's requirements. We call this approach *Dominant Network Slices*. This approach ensures optimal quality-of-service and policy compliance for network traffic, provides an intuitive model for network configuration, and supports a programmatic interface for network control. In this paper we review the approach, give examples that illustrate its power, and present results reported by an independent third-party test lab evaluation of functionality and performance of a prototype implementation.**

***Index Terms*—Network routing, Boolean constraints, partial orders, 5G mobile communication networks.**

I. Introduction

Data communication has been a part of cellular networks since the *General Packet Radio Service* (GPRS) was introduced to the 2G standard in 2000 [1]. The 3G architecture introduced a progression of enhancements to the radio access network (RAN) that significantly improved data rates, resulting in dramatic increase in the subscriber base from 788 million in 2000 to 4 billion in 2008. 4G continued evolution of the RAN, called *Long Term Evolution* (LTE) and, under the title *Enhanced Packet Core* (EPC), re-worked the mobile core architecture to be fully IP based (circuit-switching was removed from the architecture). The EPC functions were re-architected to support higher data rates to meet the growing demand for data, improve user experience with more granular support for *quality of service* (QoS), reduce the complexity of the architecture to lower capital and operational costs, and enable smooth transition to the new architecture.

At the dawn of the 5G era there are a projected 13 billion mobile users worldwide, and the universal adoption of *smartphones* has fueled continuing demand for higher data speeds, increased coverage, and improved reliability. In a relatively short time the dramatic success of cellular services and smartphones has transformed mobile data services from a luxury to a necessity.

In addition to increased coverage and bandwidth, new demands are being made of modern *mobile communication networks*. A new class of *real time* services has evolved that range from online gaming, with end-to-end latency requirements in the 30-50ms range, to ultra-low latency applications such as industrial robotics, remote medical procedures, haptic feedback, drone communications, and self-driving vehicle coordination with requirements (often in terms of life-safety) of less than 1ms. Lastly, a new universe of network devices has emerged in the form of embedded sensors and mobile transceivers in everyday devices that communicate over the network, called the *Internet of Things* (IoT). Examples of IoT applications include connected homes, health monitoring, supply chain management, vehicular communications, environmental monitoring, and surveillance deployed in consumer, industrial, public works, and military environments.

Network slices are a key enabling technology articulated in the 5G architecture to address many dimensions of these new requierments. A network slice defines a logically partitioned network that provides dedicated services and network characteristics required by a network application. Similar to a *virtual private network* (VPN) that provides logical traffic and service separation among applications in a network, a network slice also offers exclusive use of network resources to ensure the needed characteristics such as high bandwidth, lower delay, enhanced security (encryption), etc.

In addition to the three usage scenarios for network slices required in the 5G standard (eMBB, URLLC, and mMTC [2]), military applications require a wide range of capabilities that are difficult to provide in today's networks, including: specialized QoS needs not met by the standardized slice types; jam and detection resistance; control of the jurisdictions transited by network traffic; strict control of access to network resources (e.g in joint operations); rapid response to threats and vulnerabilities; prioritization of critical traffic (e.g. from troops in combat); etc. Ideally, these capabilities are provided under autonomic control such that once the battle is engaged (figuratively and literally) attention can be focused on the task at hand. Dominant Network Slices provide a framework to satisfy these requirements.

Network slicing, being a new abstraction for networks, poses real implementation challenges for current networking technology. The current state-of-the-art for implementing network slices involves a combination of network technologies including Segment Routing, BGP-LS, and path computation element (PCE) controllers [1]. The requirements to be supported of network slicing can be grouped into two general classes. *Quality of Service* (QoS) requirements specify the network performance requirements of a network application in terms of bandwidth, latency, reliability, etc. *Traffic Engineering* (TE) requirements specify non-performance related characteristics of network links such as security (e.g. encryption), jurisdictional issues (for example restricting private health information to networks within the jurisdiction of a given country), network maintenance status, etc.

Segment routing support for network slices is based on the *Constrained Shortest-Path First* (CSPF) routing algorithm. For QoS requirements, CSPF computes a single path or multiple equal cost paths that minimize a specified, additive metric (e.g. based on delay, hop count, bandwidth, etc.). For TE requirements, CSPF assigns "colors" to links and interfaces in the network. The set of colors is represented by a 32 bit color bitmap. Each color represents some attribute of a link (e.g. encryption, jurisdiction, maintenance status, etc.). Given a set of constraints (expressed in terms of link colors to be included and excluded), a traditional SPF routing algorithm is run on the subset of the topology that satisfies the constraints using the specified QoS metric. Given the result of this computation, Segment Routing computes and instantiates a single path or a set of equal cost paths from a given source to each destination for each color that satisfy its constraints.

CSPF is limited in two ways. Limiting QoS support to one or many *shortest* paths in the network is painfully restrictive. For example, the requirements for video streaming (high bandwidth and high delay) and network-based telephony (low bandwidth and low delay) are almost in conflict (a high bandwidth, low delay path would satisfy both, but at a premium price when their individual needs are not that demanding). Similarly, the color-based abstraction for TE requirements of a network is limiting. The number of attributes used for defining a policy is limited to the 32 bits in the color bitmap. The attributes available for defining policies are all related to properties of links and interfaces on a path. Policies are statically defined as a part of the network configuration.

We propose a new dominant network slice routing (DNSR) architecture based on partially ordered metrics and a Boolean algebra with Boolean variables representing policy-relevant features of the network and its environment. Using Boolean expressions from this algebra, a network administrator defines traffic classes supported in the network and specifies the QoS and TE requirements of these classes. Using these constraints DNSR, in effect, computes a dominant set of paths (being those paths where no other path has better performance) for each truth assignment to the variables used in the expressions. New flows are routed over the least-congested path in this set that satisfies the flow's requirements.

To validate this architecture we developed a prototype that implements policy-based (Layer 2) switching in an SDN-based environment using the OpenFlow protocol, the Ryu open-source controller, and Linux-based Open vSwitch software switches. The prototype includes a web interface that allows users to define the supported traffic classes for a network and

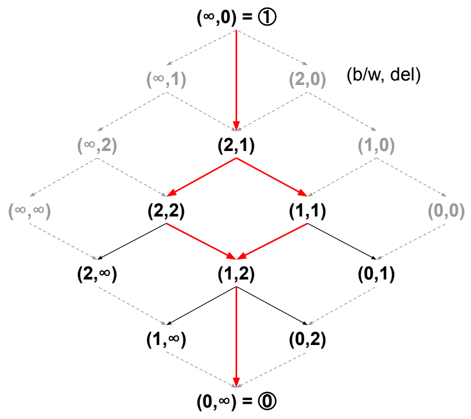


Fig. 1. Hasse Diagram of Shortest-and-Widest Constraints

the TE and QoS requirements for these classes. We engaged an independent, third-party test lab to evaluate the prototype in terms of functionality and performance.

The test lab evaluation involved 2 phases. The first phase evaluated the functionality in three different scenarios illustrating support for QoS, Zero Trust Networking (in terms of a traditional three tier web application architecture), and network segmentation of a campus network with policies regarding public, application, and security traffic classes. Functionality of the system was verified to forward traffic over paths that met the QoS and TE requirements specified for the traffic.

The second phase involved performance testing in a 4x4 torus topology. TCP and UDP traffic was transmitted between random pairs of nodes in the network, and comparison was made between DNSR and a standard network configuration based on the Rapid Spanning Tree Protocol (RSTP). For TCP, the results were that DNSR delivered up to 11% greater throughput than RSTP at *6 times the offered load*; similarly for UDP, DNSR delivered roughly equivalent goodput with close to no packet loss at up to 6 times the offered load, which was generally compatible with published simulation results [3].

II. Dominant Network Slice Routing

In previous work [3]–[6] we developed a model of routing based on partially ordered metrics and Boolean TE requirements that enhances Internet routing from the use of a single best path to every destination to a *best set of paths* that provide the full range of performance and policy available in the network. This model makes more efficient use of network resources, better meets the needs of network applications, users, and administrators, and provides a more robust default deny security model.

We proposed the use of partially ordered metrics to support the diverse performance requirements of modern network applications A *partial ordering* $(S, \succeq)$, is a set, S, with a relation $\succeq$ that is reflexive ($a \succeq a$), transitive ($a \succeq b, b \succeq c \Rightarrow a \succeq c$), and antisymmetric ($a \succeq b, b \succeq a \Rightarrow a = b$). In general, such an ordering is *partial* in the sense that not all pairs of elements in S are related ($\exists x, y \in S : x \not\succeq y,\ x \not\preceq y$); $(S, \succeq)$ is called a

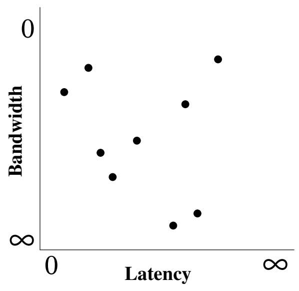


Fig. 2. Path Weights [5]

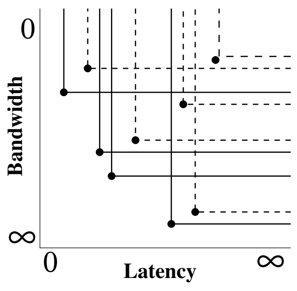


Fig. 3. QoS Regions [5]

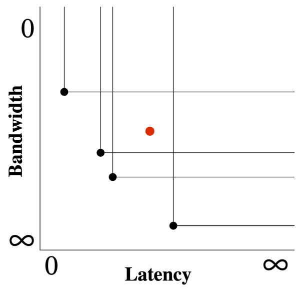


Fig. 4. Performance Classes [5]

poset. The relation $x \succeq y$ is also described as x *dominates* y, and the *dominating* subset of S is the set of elements that are not dominated by any other elements in S.

We further restrict this poset to be a bounded lattice. Informally, a poset is a lattice *iff* every pair of elements in S has both a shared ancestor and a shared descendent, and is bounded *iff* it has both a *minimum* and a *maximum* element, denoted by $\overline{0}$ and $\overline{1}$, respectively.

To help visualize lattices we use *Hasse diagrams*. The rules for drawing the Hasse diagram of a lattice are if $x \succeq y$ is in the poset then the point of y in the diagram appears below the point for x (so *lesser*, or *worse*, values are lower in the diagram), and a line is drawn between y and x if there is no z such that $x \succeq z \succeq y$.

Restricting the metrics poset to be a bounded lattice reflects the need in routing of both an ∞ and a 0 metric (corresponding, confusingly enough, to the minimum and maximum lattice elements, respectively). Also note that the $\succeq$ relation is backwards from the comparison relation used for routing ($\preceq$). In lattices the maximum element dominates all other values in the lattice. In a lattice for a routing metric, where the shortest path is best, the maximum element is the zero element of the routing metric algebra, and $0 \succeq x$ for any path weight x.

Figure 1 illustrates the use of Hasse diagrams for a metrics poset with the example of a partially ordered version of the totally-ordered Shortest-Widest path algebra (discussed in [6]), which we'll call *Shortest-and-Widest*. Weights in Shortest-and-Widest are of the form (bandwidth, delay), and $(b_1, d_1) \succeq (b_2, d_2)$ is defined as $(b_1 \geq b_2)$ *and* $(d_1 \leq d_2)$, $(b_1, d_1) + (b_2, d_2)$ is defined as $(Min(b_1, b_2), d_1 + d_2)$, $\overline{0}$ (i.e. no connectivity) is denoted by $(0, \infty)$, and $\overline{1}$ (i.e. self, or perfect) connectivity by $(\infty, 0)$. Link and path weights are combined using the $+$ operator, resulting in values that are lower in the Hasse diagram.

The red-connected subgraph is the Hasse diagram for Shortest-and-Widest where the bandwidth and delay values range over the set of values $\{0, 1, 2, \infty\}$. Note that the partial ordered nature of this definition of Shortest-and-Widest is evident here in that $(1, 1)$ and $(2, 2)$ are not comparable ($(1, 1)$ has better delay, but $(2, 2)$ has better bandwidth).

Figure 1 also illustrates an implementation issue with metric partial orders. Specifically, the black or gray-connected subset of the lattice is composed of weights that are false values in the sense that they have (exactly) one component with value 0 or ∞. These values do not reflect real world paths (any value with delay of ∞ or bandwidth of 0 is a synonym for $(0, \infty)$, and any value with the reverse, bandwidth of ∞ or delay of 0, is a variant of $(\infty, 0)$, but is not really a valid weight). The solution for these false values is to identify the ones that can result from the summing of valid weights, and have the path algebra implementation translate those values to $\overline{0}$ or $\overline{1}$, whichever is appropriate. In the figure this means translating $(2, \infty)$, $(1, \infty)$, $(0, 1)$, and $(0, 2)$ to $(0, \infty)$ (i.e. $\overline{0}$).

III. DNSR Algorithm

At their most basic, routing algorithms implement an efficient search through the paths in a network to find the *best* paths to all destinations in the network where "best" is defined based on the class of routing algorithm. Paths are typically discovered one hop at a time, starting at the source or destination (depending on the algorithm) by adding the weights assigned to each link. For partially ordered metrics the value of these weights starts with the best (maximum) value (representing the weight of a self loop for the starting node) at the top of the Hasse diagrams such as that in Figure 1, and progress down the Hasse diagram as the path is built.

Traditional shortest path routing algorithms compute routing tables with a single path to each destination. For algorithms that implement routing with partially ordered metrics best is defined as the dominant set of routes to each destination, as described in Section II, and routing tables contain a non-dominated set of paths for each destination. To illustrate this, Figure 2 plots the weights of 9 paths between a specific source and destination in an example network where the metrics are composed of bottleneck bandwidth and latency. "Better" values of these metrics are towards the origin of the graph (e.g. a perfect path would have infinite bandwidth and 0 latency).

These points can be interpreted as representing a region, up and to the right (away from the origin) of QoS values that each weight *satisfies* in the sense that the path represented by the weight would satisfy any QoS requirement in that region of the graph. Figure 3 depicts the regions satisfied by each path. Note that regions satisfied by some of the paths are fully contained in the regions of other paths. In the figure these *dominated* regions are represented with dashed lines.

A *best set* of paths to the destination can be identified as the set of paths that are not dominated by another path. This

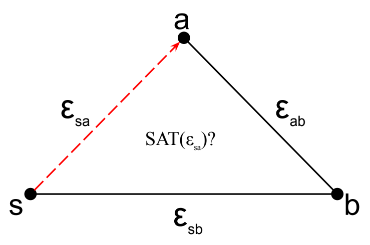


Fig. 5. Considering S-A Link

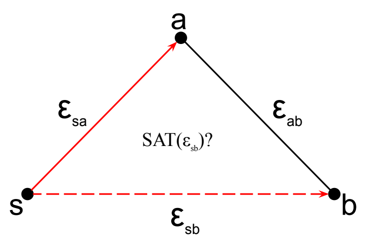


Fig. 6. Considering S-B Link

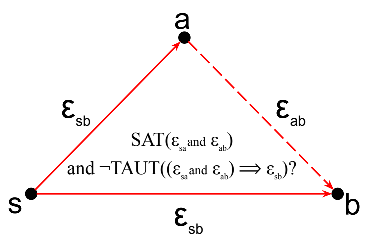


Fig. 7. Considering A-B Link

set of paths is *best* in the sense that any QoS requirement that is satisfiable by an existing path between the given source and destination, is satisfiable by a path in this set. Figure 4 shows the dominant paths for the example network, and how a given flow constraint (the red dot) can be satisfied by more than one performance class (the middle two performance classes). With this choice of paths, congestion can be minimized by selecting the least congested of the satisfying paths. Congestion can be measured by directly monitoring the link load or as a function of the flows assigned to a link; the prototype uses a simple count of the flows assigned to a link.

For routing algorithms subject to both TE and QoS requirements, best is defined as the set of non-dominated paths for each truth assignment of the Boolean variables, resulting in routing tables containing a set of paths where the path expression for each path has satisfying truth assignments not satisfied by any other path in the set with a dominating metric.

Figures 5-7 illustrate the process of building a path one hop at a time in the context of TE requirements. Each link in the network is labeled with a Boolean expression, ε, expressing the TE requirements for traffic allowed on that link. Figure 5 shows a routing computation starting at node s. The computation first considers (indicated by the dashed red line) the single-hop path from s to a. To use this path the path expression ε_{sa} must have some satisfying truth assignments, expressed by $SAT(\varepsilon_{sa})$, indicating truth assignments for traffic classes allowed on the link. Assuming the test is successful, Figure 6 shows path s-a is added to the routing table and the computation considers the next shortest path in the network, s-b, with path expression ε_{sb}. The same satisfiability test is used to add path s-b to the routing table, leading to the situation in Figure 7, where path s-a-b is considered.

This situation differs from previous iterations in that we are now considering a multihop path, and a path has already been discovered for the destination b of the new path, requiring a more complicated test to confirm that there are new satisfying truth assignments covered by the new path expression ε_{sab} to justify adding this longer path to the routing table. The figure shows how multihop paths are constructed by and'ing together the link constraints of each hop in the path; so the path expression for s-a-b is $\varepsilon_{sa}\ and\ \varepsilon_{ab}$. As previously, the $SAT(\ldots)$ test confirms there are satisfying truth assignments for $\varepsilon_{sa}\ and\ \varepsilon_{ab}$, and the additional test for tautology ($TAUT(\ldots)$, indicating *all* truth assignments are satisfied) is used to determine if there

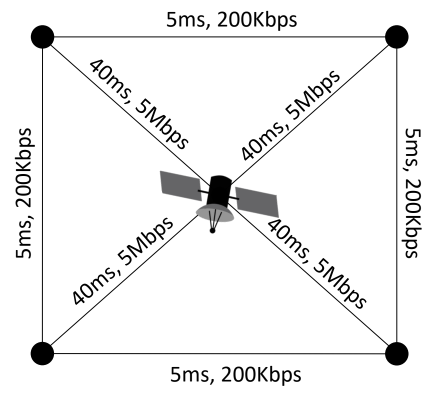


Fig. 8. Quality of Service (QoS) Scenario

are any truth assignments that satisfy $\varepsilon_{sa}\ and\ \varepsilon_{ab}$ that are not satisfied by ε_{sb}. If so, path s-a-b is added to the routing table for use by flows that satisfy those newly discovered truth assignments.

IV. Application of DNSR

The DNSR routing model presented above enables sophisticated control of network resources. In this section we present a number of scenarios to illustrate its capabilities. Each scenario is defined by the following requirements-related information. A set of *Boolean variables* representing policy-relevant attributes of flows and the network environment for use in defining the TE requirements of the network. *Link requirements* including QoS metrics of each link, and a Boolean expression defining TE requirements for the link. QoS *flow requirements* specifying a range of path metric values desired for a class of flows, where a flow class is defined by TE requirements specified using Boolean expressions. TE *network requirements*, specified by Boolean expressions, to be used in the routing function, and (with path and flow requirements) in admission control of new flows into the network. Lastly, *path requirements* are computed by the routing function using link and network requirements, resulting in a set of QoS path metrics, and a Boolean path expression defining the TE requirements of the path.

Figure 8 illustrates support of QoS requirements. The network is composed of five nodes where the four peripheral nodes are connected by low delay and low bandwidth

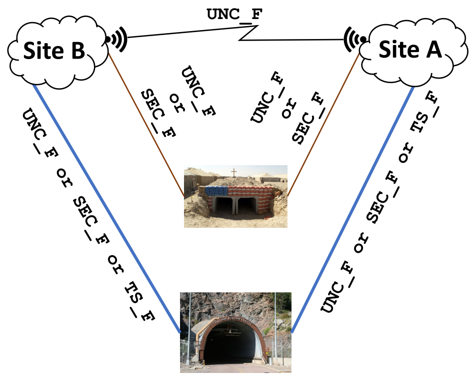
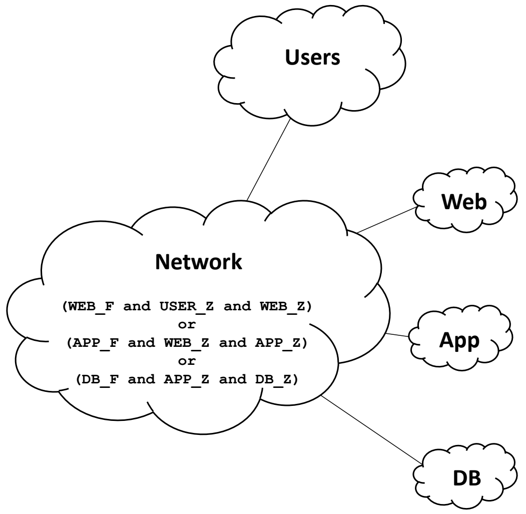


Fig. 9. Multi-Level Security (MLS) Scenario

links, and each of them is connected to a central node by high delay and bandwidth links. This can be thought of as four terrestrial nodes connected by low bandwidth terrestrial links, and one satellite node with high delay satellite links from each terrestrial node. This network is used for Voice-over-IP (internet telephone) traffic requiring low delay (for comfortable interactive conversation) for low bandwidth voice communication, and video streaming traffic, which is delay tolerant, but requires relatively high bandwidth. To support these requirements path QoS requirements are specified for `Delay` and `B/W` for both types of traffic. Since the application requirements of the network are fully expressed by the QoS requirements, there are no network TE requirements, and the traffic is forwarded over paths that satisfy each flow's QoS requirements (while minizing congestion).

Figure 9 illustrates support for multi-tenant use of networks in the form of the traditional, military-style multilevel security (MLS) model using TE requirements. In this scenario traffic is classified at unsecured (black), secret (red), or top secret (blue) security levels and is routed over infrastructure certified at the traffic's level or above. The Boolean variables `UNC_F`, `SEC_F`, `TS_F` are defined for a flow's security level. An unspecified mechanism determines the security level for a new flow, and the flow is assigned to the least congested path that satisfies the MLS routing requirement (e.g. unclassified traffic can be forwarded over paths of any security level, but top secret traffic can only traverse strongly secured paths) as specified by the TE Boolean expressions assigned to each link.

Figure 10 illustrates support for Zero Trust security applied to the traditional three layer web application architecture using TE requirements. In this architecture applications are organized into three logical tiers: web, application, and data. The web (or presentation) tier implements the user interface to the application, responsible for collecting data from the user and displaying data from the application to the user. The application (or logic) tier is where data collected from the user is processed, sometimes using information from the data tier, and results are presented to the user or saved in the data tier. The database tier is where information produced by the application is stored and managed. The benefits of



Users
Web
App
DB
Network
(WEB_F and USER_Z and WEB_Z)
or
(APP_F and WEB_Z and APP_Z)
or
(DB_F and APP_Z and DB_Z)

Fig. 10. Zero Trust Networking Scenario

this architecture include faster development, and improved scalability, reliability, and security. For security purposes, firewalls are commonly deployed between tiers.

The figure illustrates a three tier architecture implemented on a single subnet using TE requirements. Boolean variables are defined for network zones (`USER_Z`, `WEB_Z`, and `DB_Z`) and flow types (`WEB_F`, `APP_F`, `DB_F`). The zone variables could be set based on the IP prefix of servers in each zone, and some application detection technology could be used for setting the flow variables. In this scenario the links have no TE requirements, but a network-wide TE requirement is specified limiting traffic between zones to the appropriate classes of flows (e.g. `WEB_F` is only allowed between the `USER_Z` and `WEB_Z` zones, etc.). Note that, with this solution, the integrity of the three tier architecture does not depend on the location of servers. Servers from different tiers could be connected to the same layer 2 switch and the integrity of the tiers would still be maintained.

The three previous scenarios represent *static* TE requirements in the sense that how a Boolean variables is set is specified as part of configuring TE requirements for the network. So zones in the Zero Trust scenario could be defined by an IP prefix, etc. The remaining two scenarios illustrate an important capability of Boolean expression-based configurations to dynamically define the value of variables based on attributes of the network's state or environment.

Figure 11 illustrates a simple scenario where backup traffic is only allowed to flow over a core portion of the network at night. The idea being that during the day the core portions of an organization's network are reserved for operational data and backups are only allowed to traverse peripheral networks (not shown here), or be delayed to run at night. Two Boolean variables are defined including `BACKUP` which is set to true for flows that carry backup traffic, and `NIGHT` which is set to true when it is currently nighttime. The link expression (`not BACKUP or` (`NIGHT and BACKUP`)) is defined for all

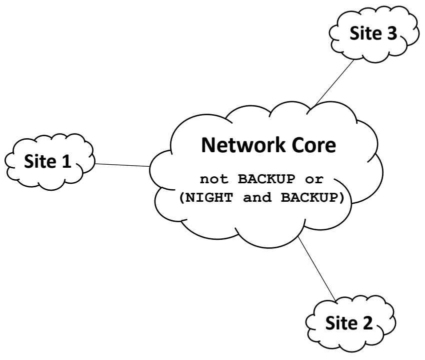


Fig. 11. Time-of-Day Backup Scenario

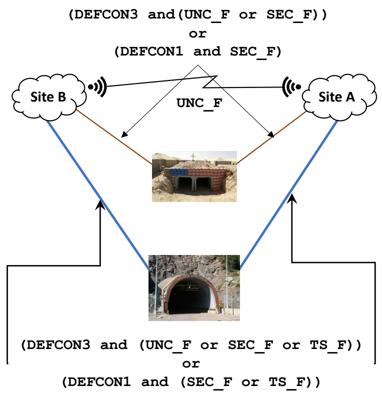

(DEFCON3 and(UNC_F or SEC_F))
or
(DEFCON1 and SEC_F)
Site B
UNC_F
Site A
(DEFCON3 and (UNC_F or SEC_F or TS_F))
or
(DEFCON1 and (SEC_F or TS_F))

Fig. 12. MLS with DEFCON Threat Levels Scenario

core network links specifying that `BACKUP` traffic can only traverse core links at night.

The Boolean variable `NIGHT` is a *dynamic* variable whose value is determined by the network at the time the flow is processed. While the time period to define as night would be configured statically as part of the network configuration, the value of the variable is determined dynamically. This capability introduces a bit of autonomic control into the network configuration, and leads to the more general solution presented next. The primary limitation to the dynamic nature of Boolean variables like `NIGHT` is they only support state directly available to the network device implementing the routing function (a router, switch, or controller).

The final example illustrated in Figure 12 implements functionality that can demonstrate a fully dynamic Boolean variable. The DEFCON/MLS scenario builds on the MLS scenario (Figure 9) by adding Boolean variables (`DEFCON1` and `DEFCON3`) reflecting the military *defense readiness condition* (DEFCON) levels used to characterize the current threat level faced by the US military. Higher threat levels are indicated by lower DEFCON numbers (with DEFCON1 indicating active nuclear war). In this scenario the MLS link expressions have been modified to integrate `DEFCON1` and `DEFCON3` threat levels. In the modified expressions, `DEFCON3` enables TE requirements equivalent to the MLS scenario (flows at a given sensitivity level are allowed to traverse links at that same level or above), but `DEFCON1` enables TE requirements that drop `UNC_F` traffic from links rated at `SECRET` and `TOP SECRET` levels. The logic being that, in a time of heightened threat, secured network resources should be reserved for important traffic.

The dynamic nature of this scenario comes from the ability to implement programmatic control of the `DEFCON` variables. In our prototype, implemented as an SDN controller with a web user interface, we implemented programmatic control as a REST service for setting the values of Boolean variables, which support the remote invocation of functions on the Web server using HTTPS messages. Using such programmatic control mechanisms, Boolean variables can be defined to reflect any state in the network or its environment that has policy significance for the network's configuration. With such variables the network's configuration can be changed immediately, without the need for reconfiguration of network devices or reprogramming of SDN-based systems.

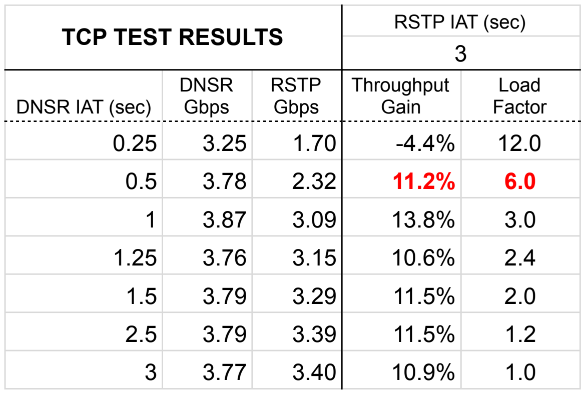
This capability has profound implications for network management. Imagine a scenario where Boolean variables are defined to reflect workstation configuration acquired using network access control technology (e.g. operating system version and patch levels) combined with variables defined to represent information from threat feeds reflecting the severity of vulnerabilities discovered in operating system versions and patch levels. TE requirements could be defined that only allowed systems to access sensitive parts of a network if they are at patch levels with no known vulnerabilities and traffic from vulnerable systems can be routed to sites that facilitate upgrades of vulnerable systems), with new vulnerabilities being integrated into network behavior as soon as they are discovered.

Lastly we observe that network slices built using DNSR enable new opportunities not easily implemented with current technology. Cross-layer coordination is naturally supported. For example special physical layer network properties, such as jam-resistance and low detectability, are important for some military applications. With DNSR these capabilities can be exposed to link or network-layer routing through the Boolean variable abstraction, and then used in defining policies that require the use of paths with these capabilities for appropriate traffic flows.

Also, as we showed with the backup traffic and DEFCON/MLS examples, the Boolean variable abstraction for TE requirements provides a powerful mechanism for implementing programmatic control of currently active network policies, enabling autonomic control of network services at time frames far shorter than is possible with humans in the control loop.

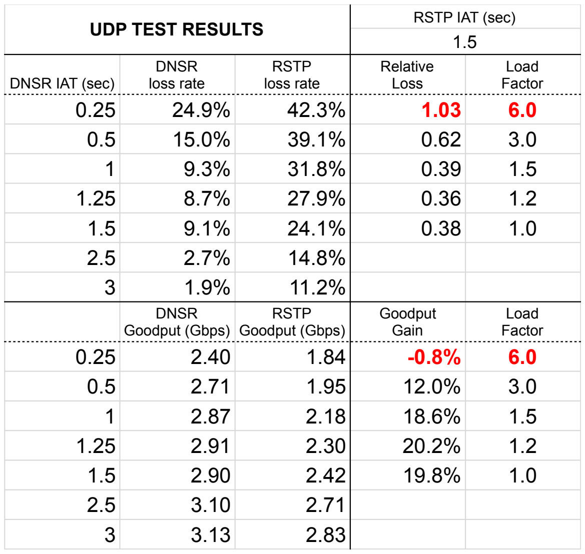
V. CNLabs Prototype Performance Evaluation

As mentioned earlier, we developed a policy-based switching prototype implementation of the architecture and submitted it to the CNLABs [7] commercial testing lab for evaluation. Focusing on the performance evaluation, they deployed the system as a 4x4 torus, with two hosts per switch, in a



TCP TEST RESULTS			RSTP IAT (sec)	
			3	
DNSR IAT (sec)	DNSR Gbps	RSTP Gbps	Throughput Gain	Load Factor
0.25	3.25	1.70	-4.4%	12.0
0.5	3.78	2.32	**11.2%**	**6.0**
1	3.87	3.09	13.8%	3.0
1.25	3.76	3.15	10.6%	2.4
1.5	3.79	3.29	11.5%	2.0
2.5	3.79	3.39	11.5%	1.2
3	3.77	3.40	10.9%	1.0

Fig. 13. TCP Performance Results



UDP TEST RESULTS			RSTP IAT (sec)	
			1.5	
DNSR IAT (sec)	DNSR loss rate	RSTP loss rate	Relative Loss	Load Factor
0.25	24.9%	42.3%	**1.03**	**6.0**
0.5	15.0%	39.1%	0.62	3.0
1	9.3%	31.8%	0.39	1.5
1.25	8.7%	27.9%	0.36	1.2
1.5	9.1%	24.1%	0.38	1.0
2.5	2.7%	14.8%		
3	1.9%	11.2%		
	DNSR Goodput (Gbps)	RSTP Goodput (Gbps)	Goodput Gain	Load Factor
0.25	2.40	1.84	**-0.8%**	**6.0**
0.5	2.71	1.95	12.0%	3.0
1	2.87	2.18	18.6%	1.5
1.25	2.91	2.30	20.2%	1.2
1.5	2.90	2.42	19.8%	1.0
2.5	3.10	2.71		
3	3.13	2.83		

Fig. 14. UDP Performance Results

VMware-based virtual environment. Each test involved 10 traffic flows for each host between random nodes in the graph with restrictions on the distribution of hops traversed (2 flows traversed 1 hop, 3 flows 2 hops, 4 flows 3 hops, and 1 flow 4 hops). Tests were run for a range of flow interarrival times (IATs) between hosts (0.25, 0.5, 1, 1.25, 1.5, 2.5, and 3 seconds). TCP performance was characterized by the cumulative throughput of all 320 flows, and UDP by the average loss rate and cumulative good-put of the flows. The relevant results are presented in Tables 13 and 14. For TCP, DNSR at 0.5sec IAT provides 11.2% better throughput at six times the load of RSTP at 3sec IAT. For UDP, DNSR at 0.25sec IAT provides roughly the same loss rate and good-put at six times the load of RSTP at 1.5sec IAT.

VI. Conclusion

We have given an overview of using partially ordered metrics and expressing TE requirements using a Boolean algebra as a powerful abstraction for implementing 5G network slices. Using these abstractions enhances network routing to compute a *best set of routes* that satisfy the full range of QoS metrics and TE requirements supported by a given network environment. Using examples, we have shown that DNSR provides a number of significant benefits.

Articulating and enforcing the QoS and TE requirements enhances the Internet's original default-allow security model to default-deny where only requirement-compliant flows are allowed. Security is further enhanced by a dramatic reduction in the network's attack surface as it is limited to network devices whose access is typically tightly controlled (compared to the attack surface of all connected devices).

Use of DNSR optimizes the user's experience, ensuring that traffic is forwarded over paths customized to the application's QoS and TE requirements and is compliant with network administration's policies. By working with a *set* of candidate paths, traffic can be forward over the least congested requirement-compliant path, dramatically improving network utilization. Simulations predicted a ten-fold increase with a somewhat "meshy" (average node degree of four) network topology [3]; these results have been verified by an independent testing lab using an un-tuned *prototype* implementation.

Network services can be safely reconfigured with programmatic control of TE Boolean variables as they do not require reconfiguration of network equipment or re-programming of software-defined networking functions. Many functions currently implemented by expensive devices external to the core network, such as firewalls, load balancers and zero-trust network equipment, can be replaced by a DNSR software upgrade. Furthermore, implementing these functions using DNSR results in significantly more robust services as they are implemented in the network layer where they have full knowledge of the network's topology.

Most importantly for many environments, DNSR provides a more intuitive, high-level network configuration paradigm based on specifying *what* the requirements of the network are, and allowing the network to solve the problem of *how* to enforce the requirements rather than depending on highly trained network engineers. This enables support of significantly more sophisticated network services by available engineers.

References

[1] S. F. Hassan, A. Orel, and K. Islam, *A Network Architect's Guide to 5G*, M. Taub, N. Davis, S. Schroeder, S. Schroeder, and B. Reed, Eds. Addison-Wesley Professional, Jun. 2022.

[2] M. Series, "IMT Vision–Framework and overall objectives of the future development of IMT for 2020 and beyond," *Recommendation ITU*, vol. 2083, no. 0, 2015.

[3] B. R. Smith and L. Thurlow, "Practical multipath load balancing with QoS," in *Proceedings International Conference on Computing, Networking and Communications*, Dec. 2013, pp. 937–943.

[4] B. R. Smith, "Efficient Policy-Based routing in the internet," Ph.D. dissertation, University of California, Santa Cruz, Aug. 2003.

[5] B. R. Smith and J. J. Garcia-Luna-Aceves, "Best-Effort Quality-of-Service," in *2008 Proceedings of 17th International Conference on Computer Communications and Networks*, Aug. 2008.

[6] B. R. Smith and J. T. Samson, "Herding packets: Properties needed of metrics for loop-free & best forwarding paths," in *2017 International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2017, pp. 408–414.

[7] CNLABS. [Online]. Available: https://cnlabs.in