Beyond Best Effort: Routing with Partially Ordered Requirements

Bradley R. Smith
Department of Computer Science & Engineering
University of California Santa Cruz
Santa Cruz, California, USA
e-mail: brad@soe.ucsc.edu

Paul S. Tatarsky
Tatarsky.com
Washington, D.C., USA
e-mail: paul@tatarsky.com

***Abstract*—Originally designed for the exchange of best effort traffic (email, web, etc.), the Internet had the modest requirements of best-effort service and global reachability. The resulting architecture provides robust and scalable networking, however it is insecure, does not support the performance and policy requirements of modern applications, and makes inefficient use of network resources. While mechanisms have been developed that attempt to address these limitations (firewalls, Policy-Routing, Traffic Engineering with Multi-Protocol Label Switching, Segment Routing, etc.), they are expensive (requiring additional devices and expensive expertise), complicated to configure, and fragile in the context of a changing network. We have developed a new routing architecture based on partially ordered routing metrics. The existing architecture assumes totally ordered metrics where, for any pair of metrics either one is better than the other or they are equal, resulting in a single "*best*" metric for any source/destination pair. Partial orders introduce the possibility of two metrics being *incomparable*. With partial orders, a *best set of (incomparable) paths* is computed between a source and destination that supports the full range of performance and policy available in the network. Traffic is forwarded over the path in this set that meets the flow's requirements and minimizes congestion. This forwarding model is compatible with the Internet's infrastructure. We have developed a prototype and submitted it to an independent testing lab that has verified the functionality and quantified the increase in performance in their testbed network.**

Keywords-Network Routing; Partial Orders; Routing Requirements; Quality-of-Service; Traffic Engineering.

I. Introduction

In this paper we enhance [1] by, in addition to presenting a new routing architecture and results from an independent lab evaluation of a prototype, also exploring *how* this new architecture is implemented using partial orders. The Internet is based on a best-effort communication model where "the network makes no specific commitments about transfer characteristics, such as speed, delays, jitter, or loss. It is assumed that end-system software, both transport layer protocols and applications, would (and must) take this unpredictability into account" [2]. Combined with reliable delivery provided by the Transmission Control Protocol (TCP) transport protocol, best-effort services provide flow-rate fairness, which is defined by the goal of equal flow rates for different flows over the same path. Flow rate fairness is an appropriate goal for best effort traffic (file transfer, email, web, etc.) [3]. As a result, the best-effort service model was a good match for the traffic the Internet was originally designed to carry. "The best-effort paradigm was very powerful - it meant that a wide range of communication technologies could be incorporated into the Internet, technologies with a wide range of basic characteristics. One factor that made the Internet protocols a success was that they could work over 'anything'" [2].

In addition, the Internet adopted a model of universal connectivity. "The original design of the Internet has been described as *transparent*: what goes in comes out. The net does not observe, filter, or transform the data it carries; it is oblivious to the content of packets. This transparency may have been the single most important factor in the success of the Internet, because transparency makes it possible to deploy a new application without having to change the core of the network. On the other hand, transparency also facilitates the delivery of security attacks, viruses, and other unwelcome data." [2]. For the original environment where the network was small and there was a high degree of trust and shared context among the users, the power of universal connectivity outweighed its risk.

In the Internet, packet forwarding is implemented on a hop-by-hop basis where forwarding tables are computed independently at each router, and the forwarding decision is done on a per-packet basis. Paraphrasing [4], packets in a flow traverse a set of interconnected networks (an internet) by, at each hop, forwarding the packet to the next hop router on the path to the packet's destination, where the next hop router *is derived from the packet's destination*. This derivation of the next hop router was initially based on the single best path in terms of a distance metric, and Internet forwarding state was composed of a *single entry for each destination* in the internet giving the next-hop router on the best path to the destination. As a result, only one path is supported to any given destination, and that path is computed to optimize a single metric.

The use of single-path routing significantly compromises the ability of a network to meet the *ordered* Quality-of-Service (QoS) and *categorical* Traffic Engineering (TE) requirements of diverse applications. Single-path routing has a similarly detrimental effect on the utilization of network resources. As the load in a network increases, sending all traffic between a given source and destination over a single path tends to result in links on that path becoming congested.

The hop-by-hop style of packet forwarding used in the Internet exacerbates this problem. With destination-based forwarding each router forwards packets by matching each packet's destination address with a single entry in the router's forwarding table. This leads to the constraint that all traffic

forwarded through an intermediate router to a destination must follow the same path used by traffic sent from that router to the destination. This aggressive tendency to concentrate traffic on a subset of a network's topology causes traffic to experience congestion while usable network resources are left idle, resulting in poor utilization of network resources.

This shortest-path model has been expanded to support Equal-Cost Multi-Path (ECMP) forwarding state composed of the *set of paths* with the same (shortest) distance metric. However, ECMP is not widely utilized, and the result is still limited to the single best path *cost* to a destination. ECMP does not address the QoS or TE requirements of a flow, and only partially addresses the poor utilization of network resources.

However, as the Internet has transitioned to the role of global communication infrastructure, with paying users of more diverse and demanding applications managing increasingly sensitive information, there is a growing need to provide QoS, trust and TE control of network resources as a basic part of the architecture.

These new requirements come from the growth of two new traffic classes called real time and policy-constrained (our term). *Real time* traffic has ordered, time-based constraints for its delivery (delay, jitter, etc.). Examples include voice, video, telemetry (e.g., computer gaming) and real time trading. *Policy-constrained* traffic has categorical constraints for its delivery. Examples include disclosure requirements (sensitive traffic must be carried on eavesdrop-resistant network infrastructure [5]), jurisdictional constraints (restrict genomics data to networks operated in a specific jurisdiction), multitenant networks (a network environment shared by multiple customers), and zero-trust environments where the network is considered untrustworthy, traffic is encrypted and strict access control enforced on what traffic can be shared between which endpoints.

In the early 2000s, the Defense Advanced Research Projects Agency (DARPA) funded the "Future Generation Internet Architecture" project (aka NewArch) to answer the question "if we could now design the Internet from scratch, knowing what we know today, how would we make the basic design decisions?" [2]. The project addressed many issues with the Internet architecture and made many intriguing recommendations. Of particular interest to this paper were two recommendations; one to transition from the Internet's traditional best-effort delivery model to a model they called *trust-modulated transparency*, and another to adopt a generalized version of the routing concept of *regions* as a first-class object in the architecture.

Trust-modulated transparency generalizes the best-effort concept to empower the network to "offer a range of behavior when two (or more) nodes communicate, based on the declared wishes of those nodes. If all the endpoints request, the flow of data among them should be as transparent and unconstrained as the Internet of today. But either end should be able to require that the packets being received be checked, filtered, or constrained in ways that limit the risk of damage and limit the range of unexpected behavior."

This paper presents our solution to this single-path routing model by combining trust-modulated transparency and regions into a unified mechanism for resource allocation in the form of a routing architecture based on computing paths subject to requirements defined by users, applications, and network administrators. This architecture makes it possible to address the problems of scaling and heterogeneity in a wide range of domains including trust, and the articulation, administration, and enforcement of resource allocation policies involving QoS and other policy-constraints. The ultimate goal being to tame the challenges of scale and heterogeneity to maximize trust, user empowerment, and the effective use of network resources.

The spirit of this trust-enhanced region abstraction is not to replace the best-effort model, but to augment it. The resulting Internet will still "work over anything," however it will also allow applications to exploit special functionality when it is available on some paths, thereby ensuring the best experience, in terms of trust, QoS, and policy compliance that is possible in a network.

The existing Internet routing architecture assumes totally ordered metrics where, for any pair of metrics, either one is better than the other or they are equal, resulting in a single "*best*" metric for any source/destination pair. Partial orders introduce the possibility of two metrics being *incomparable*. With partial orders, a *best set of (incomparable) paths* is computed between a source and destination that supports the full range of performance and policy available in the network. Traffic is forwarded over the path in this set that meets the flow's requirements and minimizes congestion.

Recent works [6], [7] have explored the related issue of routing protocols that work with partial ordered metrics. Both explore distributed routing protocols for what we have called here routing over the *best set of paths*. These two works have focused on implementing this approach in Bellman-Ford routing protocols (where paths are computed from destination back to source; see solution to "Problem B" in [8]).

This paper is organized as follows. Section II reviews current solutions that have been developed and deployed in an attempt to address the problems discussed above. Section III reviews the algebraic abstraction of partial orders and describes how we apply this to network routing using both *ordered* and *categorical* (unordered) metrics. Section IV presents the challenges presented by forwarding with partial orders, and shows how these challenges can be met by various mechanisms available in existing Internet infrastructure services. Section V provides a concise overview of our requirements-based routing approach and presents a series of scenarios that illustrate the approach. Scenarios encompass Quality of Service (QoS) management, traffic engineering for multitenant networks, zero-trust networking, the utilization of Boolean variables to reflect network state evaluated at runtime, and finally, the programmatic control of Boolean variables by external systems. Section VI outlines the challenges and opportunities we have identified for this architecture. Section VII presents the outcomes obtained from an independent testing laboratory evaluation of a prototype of this model that

we implemented. Section VIII concludes by summarizing the results and drawing conclusions.

II. Current Solutions

As described in the Introduction, the Internet's best effort communications model is limited in its ability to satisfy the QoS and TE requirements of modern network applications. A number of solutions have been developed to address these limitations under the rubric of *Traffic Engineering*.

Fundamentally, TE is the ability to route traffic over paths that differ from the lowest cost paths used by best-effort routing [5]. TE mechanisms were originally developed primarily to manage network bandwidth with the goal of minimizing congestion [9]. Since their introduction, these mechanisms have been generalized to address a broader set of requirements, such as meeting QoS requirements (specifically bandwidth and delay), restricting specific classes of traffic to topological regions of a network (i.e., multitenant capabilities), enforcing flow priorities (in the sense of preemption), and meeting administrative goals (e.g., restricting sensitive traffic to paths composed of eavesdrop-resistant media such as fiber).

This section reviews the two generations of TE technology developed to date: MPLS-TE and Segment Routing.

A. MPLS TE

The first comprehensive solution for these issues was called *MPLS TE* (Traffic Engineering with Multi-Protocol Label Switching). On its own, MPLS provides the capability to forward traffic over multiple paths, including paths that are different from the lowest cost paths used by the default best-effort routing, as required for traffic engineering. Using MPLS TE, real-time and policy-constrained traffic can be forwarded over paths that better meet their requirements and, sometimes as a specific goal and sometimes as a side-effect, distribute traffic more broadly over a network, resulting in a reduction in congestion and more efficient use of network resource.

MPLS TE accomplishes this by including additional link attributes in the routing computation, using an enhanced routing algorithm called *Constrained Shortest Path First* (CSPF) [10], and using MPLS forwarding state to forward traffic over diverse paths. In addition to the cost used in best-effort routing, MPLS TE includes additional link information such as a TE metric (distinct from the standard link cost), bandwidth, and administrative "*color*" attributes [5], [11].

For QoS requirements, CSPF computes a single path that minimizes a specified, additive metric (the traditional cost metric and an additional *TE metric*, which enables "engineering" the routing computation). For policy 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, link media (optical, copper, wireless), service-level agreement (Gold, Silver, Bronze), 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.

CSPF is limited in a number of ways. Limiting QoS support to one *least cost* path is painfully restrictive. For example, the requirements for video streaming (high bandwidth and delay tolerant; watching a movie online, a few seconds to startup is tolerated while pauses during the movie are not) 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 flow 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.

MPLS-TE implements point-to-point (P2P) forwarding state specific to each flow, resulting in very poor utilization of label-swap resources and poor scalability. Lastly, MPLS-TE implements on-demand route computation and path signaling, adding significant overhead to the forwarding process.

These limitations led to the development of the improved Segment Routing architecture.

B. Segment Routing

A more recent solution for the original Internet architecture's limitations involves a combination of network technologies based on Segment Routing (SR) [12]. SR computes and builds paths similar to MPLS-TE that better meet the QoS and TE needs of network applications. When TE is not required, SR is able to implement ECMP paths.

SR improves on MPLS-TE in a number of ways. SR integrates the label distribution, TE path signaling, and routing functions that are implemented separately in MPLS-TE into a single protocol. SR builds any-to-one, "multi-point to point" (MP2P) label-swap forwarding state. SR implements a forwarding model that still includes an on-demand routing computation, but makes use of pre-computed forwarding state. The resulting solution is dramatically simpler to configure and operate than MPLS-TE, much more efficient in its use of label-swap resources, and improves on the MPLS-TE forwarding process.

While SR improves on MPLS-TE in the ways listed above, it inherits some of MPLS-TE's limitations including only supporting least-cost paths, its use of the limited abstraction of colors for TE requirements, and it still requires a routing computation for each new flow.

III. Partial Orders

With its use of single-path routing the Internet adopted a routing architecture based on a total ordering of paths available in the network between a given source and destination. Based on this ordering, a single path is used to send traffic to the destination. As discussed in the Introduction, for a small network where there was a high degree of trust, a set of

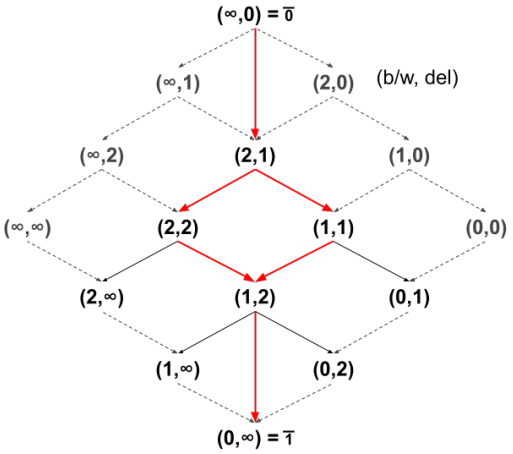

Figure 1. Hasse Diagram for Shortest-and-Widest.

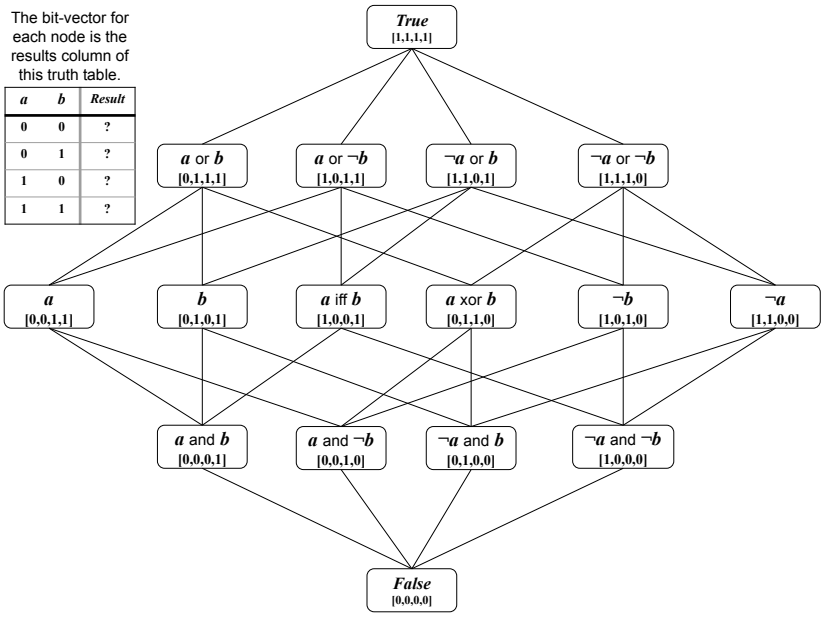

Figure 2. Hasse Diagram of Boolean Constraints on 2 Variables

applications with similar requirements of the network, and minimal requirements for efficient use of network resources, this single-path model was an appropriate choice. However, for the current and future Internet, with the highly diverse requirements and high priority for efficient use of network resources that come with being the converged network infrastructure for the twenty-first century, the single-path limitation is no longer acceptable. Our solution generalizes this architecture to implement a model based on *partial orders* that natively supports the use of diverse paths to a destination that provide the full range of services available in a network and are used based on the requirements of the application generating a given network flow.

A *partial order* $(S, \preceq)$ is a set S with an *order relation* $\preceq$ that is reflexive ($a \preceq a$), transitive ($a \preceq b, b \preceq c \Rightarrow a \preceq c$), and antisymmetric ($a \preceq b, b \preceq a \Rightarrow a = b$). For network routing S is composed of the weights of links and paths in a network, and $\preceq$ defines an order on these paths that is used in the routing computation to identify the set of paths described above.

In general, such an ordering is *partial* in the sense that not all pairs of elements in S are related ($\exists x, y \in S : x \not\preceq y,\ x \not\succeq y$); therefore $(S, \preceq)$ is called a *partially-ordered set* (or *poset*). In *partially-ordered constraint optimization* [13], the relation $x \preceq y$ among constraints is also called $dominates$, and the $dominating$ subset of S (i.e., the set of elements that are not dominated by any other elements in S) is called the *Pareto frontier*. The special case of an ordered set where $\forall x, y \in S$ either $x \preceq y$ or $x \succeq y$ is called a *total order*.

Network routing uses a special kind of partial order called a *lattice*. A lattice is a partial order where every pair of elements x, y in S have both a shared ancestor ($z \in S : z \preceq x, z \preceq y$) and descendant ($w \in S : x \preceq w, y \preceq w$) [14]. These properties are required for network routing as every pair of path weights are guaranteed to have this property. Specifically, all path weights have a guaranteed shared ancestor with the weight of an ideal path (typically used to denote the path from a node to itself), shown as $\overline{0}$ in Figure 1, and shared descendant with the weight of a non-existent path (typically used for the weight between unconnected nodes), shown as $\overline{1}$.

NOTE: order relations are normally defined with the $\succeq$ relation, where $a \succeq b$ is interpreted as a is *better than or equal to* b; however, in network routing, *smaller* (i.e., *shorter*) is considered better, so in this paper we use the network routing convention of $a \preceq b$, and label the top and bottom nodes $\overline{0}$ and $\overline{1}$, respectively (vs the standard notation of $\overline{1}$ and $\overline{0}$).

To help visualize lattices we use *Hasse diagrams*. The rules for drawing the Hasse diagram of a lattice are if $x \preceq y$ is in the poset then y appears below x in the diagram (so a path with weight x is *better* than, or *dominates*, a path with weight y), and a line is drawn between y and x if there is no intermediate value z such that $x \preceq z \preceq y$.

Figure 1 illustrates the use of lattices and Hasse diagrams for routing with the example of a partially ordered version of the Shortest-Widest path algebra, which we'll call *Shortest-and-Widest* to distinguish it from the totally ordered version (discussed in [15]). Weights in Shortest-and-Widest are of the form (bandwidth, delay), and $(b_1, d_1) \preceq (b_2, d_2)$ is defined as $(b_1 \geq b_2)\ and\ (d_1 \leq d_2)$, $(b_1, d_1) \oplus (b_2, d_2)$ ($\oplus$ is used here to distinguish it from the $+$ used in the following) is defined as $(Min(b_1, b_2), d_1 + d_2)$, $\overline{0}$ (i.e., self, or perfect) connectivity by $(\infty, 0)$, and $\overline{1}$ (i.e., no connectivity) is denoted by $(0, \infty)$.

Figure 2 illustrates the partial ordered nature of Boolean constraints with a Hasse diagram for expressions of two

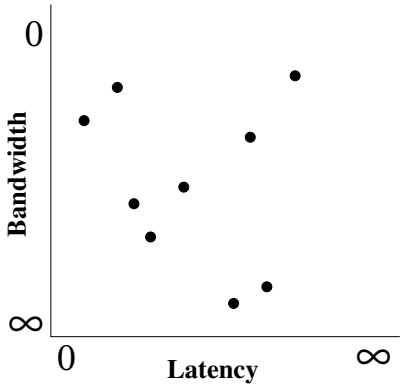

Figure 3. Path Weights

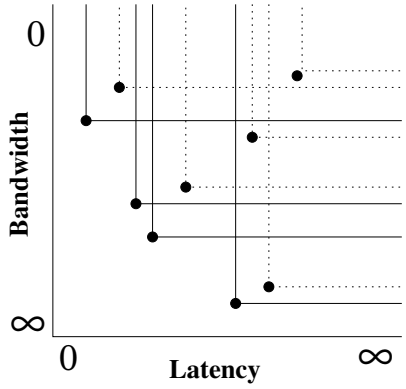

Figure 4. QoS Regions

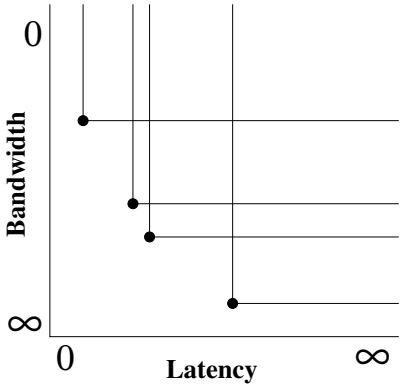

Figure 5. Best Set

Boolean variables. Boolean constraints provide a powerful abstraction for expressing *categorical* (i.e., unordered) resource utilization policies. Path comparison ($\preceq$) is defined as the partial order *dominates* relation where $\varepsilon_1 \preceq \varepsilon_2$ (i.e., ε_1 *dominates* ε_2) when ε_1 is $True$ for every truth assignment where ε_2 is $True$. Formally, this is true when $\varepsilon_2 \Rightarrow \varepsilon_1$ is a tautology (i.e., is always $True$). Path construction ($\oplus$) is defined as the conjunction (and'ing) of two expressions, resulting in values that are generally lower in the diagram (e.g., $(a \texttt{ or } b) \oplus (a \texttt{ or } \neg b) = a$).

Figures 3 through 5 give an intuitive sense of the partial-ordered nature of Shortest-and-Widest. Figure 3 plots the weights of 9 paths between a specific source and destination in an example network where the metrics composing the weights are bottleneck bandwidth and latency. "Better" values of these metrics are towards the origin of the graph (i.e., 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 4 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 is defined as the dominating set of the weights of paths to the destination. This set of paths is *best* in the sense that any bandwidth and delay requirements that are satisfiable by an existing path between the source and destination, are satisfiable by a path in this set. Figure 5 shows the best set of routes for the example network.

The red-connected subgraph in Figure 1 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).

Lastly, the *best* metrics (from a routing sense) are towards the top of the Hasse diagram (e.g., $(\infty, 0)$, ∞ bandwidth and 0 delay, is the value for a self-loop in Shortest-and-Widest) and the *worse* metrics are towards the bottom (e.g., $(0, \infty)$, 0 bandwidth and ∞ delay, is the value for unreachable in Shortest-and-Widest), reflecting the partial ordering semantics for the $x \preceq y$ relation being that x is better than y.

Figure 1 also illustrates an implementation issue with 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). However, given the constraint that real link/path weights should have neither component with a 0 or ∞ value, only some of these "false" weights can occur in the normal course of a routing algorithm. Specifically, in this Shortest-and-Widest example, the false values connected to the graph by solid arrows can occur in a real computation. E.g $(2, 2) \oplus (2, 1) = (Min(2, 2), 2 + 1) = (2, \infty)$. 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., $\overline{1}$).

IV. Forwarding with Partial Orders

In a hop-by-hop routing environment we make a distinction between a path and a *forwarding path*. In hop-by-hop routing, the shortest paths to each destination are computed for each node, and a forwarding table composed of the next hops on these paths is installed at the node. As a packet travels through the network the node at each hop independently selects the next hop based on the forwarding table computed at that node. Forwarding paths are an emergent property of the collective routing tables of all nodes, are not known by any single node, and may be different from paths computed at any given node.

All previous work on path algebras has focused on identifying the properties needed to ensure forwarding paths resulting from use of a path algebra are loop-free and best paths. These properties were appropriate for the implicit focus on totally

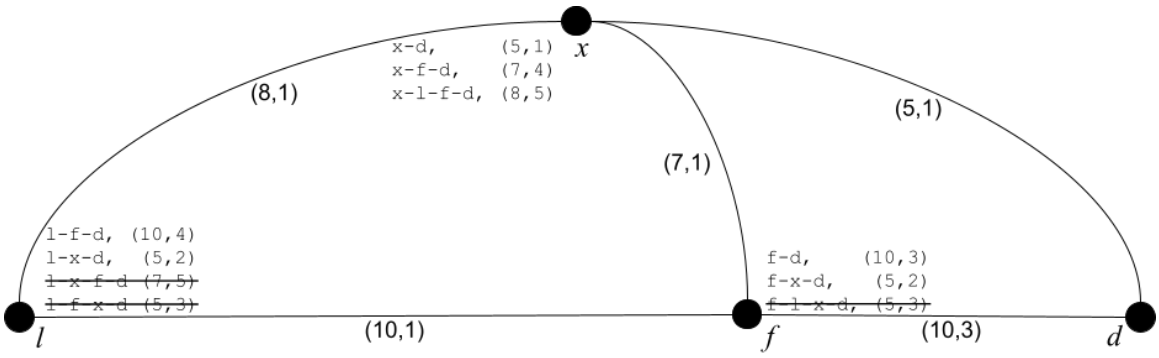

Figure 6. Shortest-and-Widest Scenario.

ordered path algebras where there was guaranteed to be a *single best* path weight (of possibly multiple equal cost paths) for each destination.

However, with the focus here on partially ordered path algebras, this guarantee is no longer valid. In general there will be a set of paths with different weights to each destination, and the concept of best will not apply. The generalized definition we use for *well-behaved* routing and forwarding with partially ordered path algebras is that *forwarding paths are loop free and have the same weight as the weight of the path selected at the source*. Note, this definition works equally well for the totally ordered case, which is appropriate.

Similarly, previous work on totally ordered path algebras has assumed a simplified forwarding model where, at each hop, traffic is forwarded to the next hop on the best path computed by the routing function. Again, in a multi-path environment, this no longer makes sense. With multiple paths to each destination having different path weights, each hop will have to make a choice of the available paths to use in forwarding traffic. Therefore we need to define a generalized forwarding model to be used with the best set of routes computed using a partially ordered path algebra. First we present a simple generalization of the single-path forwarding model and show that it does not work for a path algebra (Shortest-and-Widest) that, intuitively, seems like it should be well-behaved.

With traditional routing, decisions are made at each hop to forward traffic along the best path to the destination from the current node. Given the multiple incomparable paths available with partial orders, this use of a *best* path no longer makes sense and a process along the lines of the following must be used:

> Given a destination and a performance constraint (PC), at each hop choose a path from the set of best paths computed for the destination where the weight of the forwarding path from source to destination satisfies PC. Specifically, given the weight of the path taken to reach this hop (call this the preceding path weight, or PPW), choose a path from the current node's dominant route set where the path's weight (succeeding path weight, or SPW) is such that $PPW \oplus SPW \succeq PC$.

This process is most like traditional forwarding in that decisions are made at each hop on a packet-by-packet basis. The problem with this decision process is it can loop in the context of the Shortest-and-Widest algebra, as shown in Figure 6. In the figure the dominant route set for destination d is shown at each router, and includes the path and path weight (given as *(bandwidth, delay)*). The figure shows the full list of loop-free paths available at each router, with those routes not in the dominating set ~~crossed out~~ to show how the dominant set is selected. Specifically, at node l, $l-f-d \prec l-x-f-d$ (i.e., $(10,4) \prec (7,5)$) and $l-x-d \prec l-f-x-d$ (i.e., $(5,2) \prec (5,3)$.

Using this topology with traditional forwarding there are many opportunities for loops. For example, consider node l forwarding a flow with a PC of $(4,8)$; l could forward it to either node x ($(5,2) \preceq (4,8)$) or node f ($(10,4 \preceq (4,8)$). Both could forward to the other, resulting in a possible loop; e.g., x could forward to f ($(8,1) \oplus (7,4) \preceq (4,8)$), and f could forward back to x ($(8,1) \oplus (7,1) \oplus (5,2) \preceq (4,8)$) until eventually the delay of the looping path gets too great and one of x or f is forced to either forward to d directly or drop the flow (specifically, after 4 trips over the $x-f$ link node x would see that none of its routes satisfy the constraint described above (e.g., $(8,1) \oplus (7,1) \oplus (7,1) \oplus (7,1) \oplus (7,1) \oplus (7,4) = (7,9) \not\preceq (4,8)$), and it would be forced to forward directly ($(8,1) \oplus (7,1) \oplus (7,1) \oplus (7,1) \oplus (7,1) \oplus (5,1) = (5,6) \preceq (4,8)$).

The result is traffic being forwarded over a path that wastes network resource by traversing segment $x-f$ four times and a path that is worse than the options available at node l. This problem comes from the extra degree of freedom offered by the *best set* of routes available at each node. There needs to be some mechanism for communicating the decision made at the ingress router to subsequent routers to ensure these loops are avoided along the lines of the following:

> Given a destination and the full path weight (FPW) selected at the source (where FPW satisfies the flow's performance constraint), at each hop choose a path from the set of paths computed for the destination such that the weight of the forwarding path from

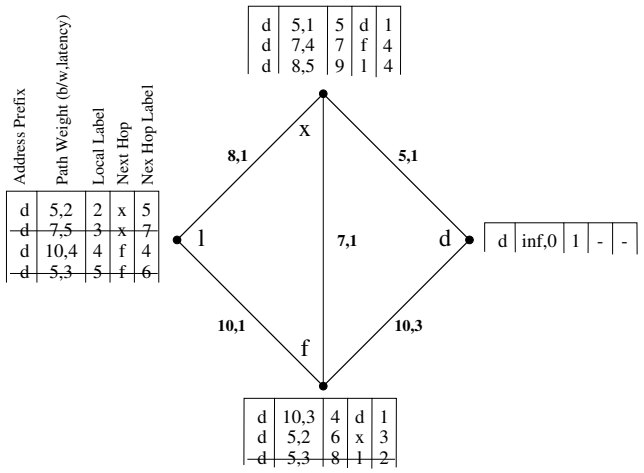

Figure 7. Shortest-and-Widest with Labels for Destination d.

source to destination equals FPW. Specifically, given PPW and SPW described above, choose a path from the current node's dominant route set such that $PPW \otimes SPW = FPW$.

This eliminates the problems experienced by the process described above. Specifically, referring to Figure 6 again, if l forwards the traffic to x, x is forced to forward the traffic directly to d because only route $x - d$ at x satisfies the constraint that $(8,1) \otimes (5,1) = (5,2)$.

While this may initially appear to be an expensive forwarding model (due to the need to communicate FPW and PPW to each hop), it can actually be efficiently implemented in current networks using existing technology that causes the ingress router's forwarding decision to fix the forwarding decisions of each hop along the path. The conceptually simplest solution, in the context of a full-topology routing environment (e.g., link-state routing where each router has a full picture of the network topology) would be the use of source routing (see [16] for a description of source routing in the context of specific requirements).

More recently, segment routing [17] offers a similar solution. Both of these solutions work with the full path to the destination, and therefore the existence of a forwarding path is guaranteed (barring topology changes, which would trigger a re-computation of paths and an update of source/segment routes).

MPLS [18] provides a more general solution, that would support routing environments involving less topology information including, in the extreme, distance-vector routing (where only next hops are known by each router). The MPLS solution works by neighboring routers exchanging label information following route re-computations. Given this mechanism, MPLS follows a path equivalent to the path selected by the ingress router (where there are multiple equal-cost paths, the path selected by the ingress router and the forwarding path may be different paths, but will both be members of this set of equal cost paths).

With label-swap forwarding only the first router that handles a packet has to classify the traffic and select a path for the flow before forwarding it; all subsequent routers perform a simple label-swap lookup. Forwarding state is enhanced to include local and next hop label information, and path weight information to be used in selecting a path.

Figure 7 gives an example network configuration that reflects Figure 6. Of specific interest are the routes between nodes l and d, which include two routes for each path with weights $(5,2)$ (going through x) and $(10,4)$ (going through f) where the path through x has better latency but worse bandwidth than the path through f, illustrating the ability of MPLS to support multiple incomparable paths between two nodes. All of these solutions, which provide control of the path to be used for forwarded traffic to the source router, support the efficient use of multiple paths between a given source/destination pair.

V. Beyond Best Effort

As described in the Introduction, our requirements-based routing architecture implements the *trust-modulated transparency* and routing *region* capabilities identified by the DARPA NewArch project as needed to address the requirements of modern network applications. Specifically, routing based on partially-ordered requirements computes and forwards traffic over paths that satisfy requirements articulated by users, applications and network administrators for each flow carried in a network. As a result traffic carried in a given routing domain ("region") complies with the QoS and TE requirements defined for that domain. The result is an augmented best-effort architecture where the Internet protocols are still able to work over "anything," but now are able to exploit special functionality in the network when it is available, ensuring the best experience in terms of trust, QoS, and policy-compliance that is possible in a given region.

The rest of this section illustrates the mechanics and power of this approach with a number of scenarios. Each scenario is defined by a set of requirements for how traffic in a given class of flows is to be handled. As described in the Introduction, there are two types of requirements: QoS and TE.

QoS requirements of a network application address the *ordered*, performance requirements needed for an application to perform well, typically expressed in terms of bandwidth, latency, jitter (variation in latency), reliability, etc. *TE* requirements specify the *categorical*, 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.

A. Quality of Service

As an example, consider a network being used by both an interactive voice application implementing an Internet-based telephony service (commonly called Voice over IP, or VoIP), and a video streaming service such as Netflix.

TABLE I. VOICE/VIDEO QOS REQUIREMENTS

Flow Type	Perf Rqmts	
VoIP	$\leq$ 40ms	$\geq$ 100Kbps
Video Streaming	$\leq$ 10sec	$\geq$ 3Mbps

TABLE II. MULTI TENANT TE REQUIREMENTS

Flow Type	Boolean Variable	Path Expressions
Tenant A	TA	TA
Tenant B	TB	TB
		(TA **or** TB)
		(TA **and** TB)
		True
		False

Interactive voice communication has relatively modest bandwidth requirements (100Kbps provides a high quality voice encoding) but fairly stringent delay requirements (interactive communications is awkward with delays much above 150ms[19]). So, VoIP service requires low delay and can can live with relatively low bandwidth. In contrast, video streaming has very modest delay requirements, but relatively high bandwidth requirements (i.e., even many seconds delay in starting a video is tolerable as long as once it starts there is adequate bandwidth for it to smoothly run to completion). So, a video streaming service requires high bandwidth and can live with high delay. Table I shows these requirements.

Given these performance requirements defined in terms of delay and bandwidth, the routing computation collects topology information that includes QoS metrics for each link. It then runs a modified shortest-path first routing algorithm that computes the set of paths in the network that are not comparable to each other, and forwards traffic over one of these paths that satisfies the flow's performance requirements; in the event there are more than one it uses the least congested.

Using the video and voice example from above, the low and high bandwidth and delay paths can be see as *incomparable*. Specifically, low delay is better than high delay however high bandwidth is better than low. This incomparability can be restated as *it depends on the needs of the flow*, resulting in the opportunity to compute a *best set of routes* as those paths in the network where some application might prefer one path over the others. Further, with potentially a choice of satisfying paths, it is possible to distribute traffic more widely over a network, thereby reducing congestion and increasing utilization.

B. Multitenant

Multi-tenancy is when several network customers are sharing a set of network resources, such as when several different small business are using the same network resources to communicate within their offices in a building and to reach the Internet. Despite the fact that they share resources, these network customers are not aware of each other, and their data is typically kept separate.

TABLE III. ZERO TRUST

Flow Types	Zones	End-to-End Requirements
$\texttt{WEB}_F$	$\texttt{USER}_Z$	($\texttt{WEB}_F$ **and** $\texttt{USER}_Z$ **and** $\texttt{WEB}_Z$)
$\texttt{APP}_F$	$\texttt{WEB}_Z$	($\texttt{APP}_F$ **and** $\texttt{WEB}_Z$ **and** $\texttt{APP}_Z$)
$\texttt{DB}_F$	$\texttt{APP}_Z$	($\texttt{DB}_F$ **and** $\texttt{APP}_Z$ **and** $\texttt{DB}_Z$)
	$\texttt{DB}_Z$	

To implement such a set of requirements we define a set of Boolean variables that reflect policy-relevant attributes of network traffic, the network itself, or of the network's environment. TE requirements are articulated as Boolean expressions composed of these variables, and are used in the routing computation to compute policy-compliant paths for the flow to use.

A subset of these expressions can be used to label links in the network to express the TE constraints each link imposes on traffic that traverses the link. Path expressions are constructed as a part of the routing computation (by *and*'ing together the link expressions), to express the constraints imposed on traffic that traverses the path. Expressions that are not assigned to links define what we will call *end-to-end* requirements that are used to define requirements of traffic in terms of its content, source, and destination. We will see examples of all of these in the following.

Table II shows the Boolean variables that could be defined to support two tenants, and some likely path expressions that would be used to control traffic on a multitenant network. The Boolean expressions extracted from a flow are used to determine if a flow can use a path by determining if the conjunction (and'ing) of the flow expression with the path's expression is *satisfiable* (meaning there is a truth assignment to the variables that results in a $True$ value for the combined expression).

The `True` and `False` path expressions indicate any or no flows may use a link, respectively (these expressions can be used for any path expression and are not included in the remaining scenarios). `TA` or `TB` represent traffic sent or received by tenant A or B (perhaps set based on a flow's source or destination address). (`TA` **or** `TB`) allows tenants A and B to share a link, and (`TA` **and** `TB`) indicates a link only for use for flows between tenant A and B.

C. Zero Trust

This scenario illustrates support for Zero Trust security applied to the traditional three layer web application architecture using TE requirements. The general Zero Trust architecture, based on the assumption that networks cannot be trusted, adopts a least privilege strategy by encrypting all traffic and strictly enforcing access control expressed as an access matrix specifying what combination of users, applications, and security zones can access other security zones. Security zones are logical containers for physical interfaces, VLANs, and IP address ranges (i.e., a region of the network) [20].

In the three layer web application architecture, applications are organized into three logical tiers: web, application, and

TABLE IV. CONTROL BACKUPS OVER CORE

Flow Types	Time Periods	Path Requirements
BKP	NT	(**not** BKP **or** (NT **and** BKP))

data. The web (or presentation) tier is 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 this architecture include faster development, and improved scalability, reliability, and security. For security purposes, firewalls are commonly deployed between tiers.

Table III illustrates a three tier architecture implemented on a single subnet using TE requirements. Boolean variables are defined for flow types ($\texttt{WEB}_F$, $\texttt{APP}_F$, $\texttt{DB}_F$) and network zones ($\texttt{USER}_Z$, $\texttt{WEB}_Z$, and $\texttt{DB}_Z$). The zone variables could be set based on the IP prefix of servers in each zone, and TCP ports or application detection technology could be used for setting the flow variables. In this scenario the links have no TE requirements, but end-to-end TE requirements limit traffic between zones to the appropriate classes of flows (e.g., $\texttt{WEB}_F$ traffic is only allowed between the $\texttt{USER}_Z$ and $\texttt{WEB}_Z$ 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 two 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.

D. Dynamic Variables

Table IV 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, or be delayed to run at night.

Two Boolean variables are defined including BKP, which is set to true for flows that carry backup traffic, and NT, which is set to true when it is currently nighttime. The link expression (**not** BKP **or** (NT **and** BKP)) is defined for all core network links specifying that BKP traffic can only traverse core links at night.

The Boolean variable NT is a *dynamic* variable whose value is determined by the network at the time the flow is processed.

TABLE V. BOOLEAN SATISFIABILITY AND ONEHOT()

DY	NT	BKP	Path Req	OH(DY,NT)	Result
$False$	$False$	$False$	$True$	$False$	$False$
$False$	$False$	$True$	$False$	$False$	$False$
$False$	$True$	$False$	$True$	$True$	$True$
$False$	$True$	$True$	$True$	$True$	$True$
$True$	$False$	$False$	$True$	$True$	$True$
$True$	$False$	$True$	$False$	$True$	$False$
$True$	$True$	$False$	$True$	$False$	$False$
$True$	$True$	$True$	$True$	$False$	$False$

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 NT is they only support state directly available to the network device implementing the routing function (a router, switch, or controller).

There are some subtleties to satisfiability that need explanation. We illustrate this by adding a variable DY that is $True$ for a flow occurring during they day (added for illustration since DY can be expressed as (**not** NT)). The first four columns of Table V show the truth table for the path expression (**not** BKP **or** (NT **and** BKP)) given these three variables. This shows that a flow sent during the day, with DY set to True and NT not set (i.e., in a "don't care" state), would be allowed because the path requirements would be satisfied in the last two rows, which is a mistake. This mistake comes from the fact that we have not expressed the requirement that a flow can only occur either *during the day or night but not both*, which is why the last two rows of the fourth column (where both DY and NT are $True$) show as $True$. To fix this we need a Boolean expression of the DY and NT variables that is $True$ only for truth assignments where only one variable is $True$. We represent such a function as $OH(variables...)$ (short for $OneHot(...)$) in the fifth column of Table V, and use it to complete the satisfiability test.

Applying this to our problem, the "Result" column shows the conjunction of the path requirements and OneHot(DY,NT) columns, where only rows three through six are valid, and show the desired truth table (the only blocked flows are backup flows not sent at night). So whenever we have a set of variables where only one can be $True$ for a given flow, we must include the $OneHot(...)$ function of those variables in the combined flow and path expression to avoid false positives. This is assumed in the examples in the paper. Note, a similar set of constraints is needed for the Zero Trust scenario.

E. Programmatically-Controlled Variables

The final example illustrated in Table VI implements functionality that can demonstrate a fully dynamic Boolean variable. This scenario has two components, DEFCON threat levels and MultiLevel Security (MLS). MLS provides support

TABLE VI. DEFCON WITH MULTILEVEL SECURITY

Flow Types	Threat Levels	Path Requirements
TS_f	D_1	((D_1 **and** (U_f **or** S_f **or** TS_f)) **or** (D_3 **and** (S_f **or** TS_f)))
S_f	D_3	((D_1 **and** (U_f **or** S_f)) **or** (D_3 **and** S_f))
U_f		(U_f)

for multitenant use of networks in the form of the traditional, military-style multilevel security using TE requirements. Traffic is classified at unsecured, secret, or top secret security levels and is routed over infrastructure certified at the traffic's level or above. The Boolean variables U_f, S_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.

DEFCON builds on MLS by adding Boolean variables (D_1 and D_3) reflecting the military *defense readiness condition* (DEFCON) levels used to characterize the current threat level. Higher threat levels are indicated by lower DEFCON numbers (DEFCON1 being the highest threat level). In this scenario link expressions include MLS and DEFCON variables. In these expressions, D_3 enables traffic handling equivalent to standard MLS policies described above (traffic is routed over infrastructure certified at the traffic's level or above), while D_1 enables policies that drop unclassified (U_f) traffic from links rated at S_f and TS_f 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 a Software-Defined Network (SDN) controller with a web user interface, we implemented programmatic control as a Representational State Transfer (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 policy enforced in a network 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.

VI. CHALLENGES AND OPPORTUNITIES

A fundamental challenge of requirements-based routing is the need to determine the *satisfiability* of Boolean expressions used to express categorical requirements [21]. Satisfiability, which is the test of whether there is a truth assignment of the variables in a Boolean expression that cause the expression to evaluate to *True*, is the prototypical NP-Complete problem [22]. The essential meaning of this is there is no known way to determine satisfiability "efficiently".

One possible approach to containing the cost of the satisfiability test is to restrict the syntax of these expressions to forms with efficient algorithms for satisfiability. Significant work has been done along this line, culminating in Schaefer's Dichotomy theorem [23]. Schaefer's theorem comprehensively defines the boundary between expressions for which satisfiability can be determined efficiently and those for which no efficient solutions are known. The theorem shows that efficient solutions exist for six classes of expressions, and any expressions not in these classes are NP-complete.

Unfortunately for the work here, Schaefer also showed that none of these classes support negation, which is required for routing with requirements. However, fortunately, driven by the needs of integrated circuit design testing, there has been dramatic progress in the optimization of satisfiability algorithms such that, in spite of the inherent challenges of the general problem (e.g., current algorithms can determine satisfiability of expressions with millions of variables and clauses in minutes [24]).

These results, and the likely size and characteristics of requirements-based routing problems, give hope that the cost of satisfiability will not be a problem. Experience with our (un-tuned and research-grade) prototype, where path selection based on Boolean requirements are made once per flow, is that the time required for these decisions is consistent with normal switching speeds (single-digit milliseconds). Additionally, we have not implemented the use of "assumptions" [25], which should significantly speed up determining satisfiability in the path selection process.

At a more engineering-level, there are a number of other challenges/opportunities that need to be addressed. Architectures for forwarding traffic over multiple paths to the same destination (currently include OpenFlow [26], P4 [27], and MPLS [5])) are in constant flux. Assessing the scalability and performance of solutions requires attention, and possibly impacts the architecture for a comprehensive solution.

Regarding opportunities, developing and assessing distributed implementations of this technology, along the lines of traditional routing protocols, needs to be evaluated as an approach to addressing scalability and performance issues.

TABLE VII. TCP PERFORMANCE RESULTS

TCP TEST RESULTS			RSTP IAT (sec) 3	
RPO IAT (sec)	RPO Gbps	RSTP Gbps	Throughput Gain	Load Factor
0.25	3.25	1.7	-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	3.4	10.9%	1.0

TABLE VIII. UDP PERFORMANCE RESULTS

UDP TEST RESULTS			RSTP IAT (sec) 1.5	
RPO IAT (sec)	RPO 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%		
	RPO Gput (Gbps)	RSTP Gput (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		

As mentioned earlier, recent work along these lines [6], [7] has explored related approaches to routing using distributed Bellman-Ford routing protocols.

VII. PROTOTYPE

To validate this architecture, we developed a prototype that implements policy-based (Layer 2) switching in a software-defined networking (SDN) environment using the OpenFlow protocol, the Ryu open-source controller, and Linux-based Open vSwitch [28] software switches. The prototype includes a web interface that allows users to define the supported traffic classes for a network and the TE and QoS requirements for these classes.

Implementation in Layer 2 was done for both convenience and functionality. A centralized, controller-based implementation made configuration significantly easier by centralizing the definition and implementation of policy in one place. Additionally, implementation of the requirements-based routing model at Layer 2 provides fine-grained control of network traffic down to the switch port level, enabling the full power of this architecture to be displayed. However, with some loss of granularity (working at the subnet vs switching level), this architecture can support a Layer 3 implementation equally well.

We engaged an independent third-party test lab to evaluate the prototype in terms of functionality and performance. Functionality testing involved evaluation of three scenarios: QoS with VoIP and video traffic (see Section V-A), a Zero Trust network environment (Section V-C), and a network segmentation environment (similar to the scenario covered in Section V-B) emulated in a small enterprise network environment.

The findings of these tests verified the expected results for the three scenarios. In the QoS configuration, routing with requirements effectively selects an appropriate path for different types of network traffic that have distinct bandwidth and delay requirements. In the Zero-Trust scenario, routing with requirements implements access control for users, applications, and network zones. And for network segmentation, routing with requirements enforces access control for different zones in the network. Furthermore, across all three scenarios, if there are multiple paths that satisfy the flow's requirements, routing with requirements selects the path with the least amount of traffic (thereby minimizing congestion).

For performance testing, they measured the performance of TCP and UDP traffic among hosts attached to switches connected in a 4x4 torus topology. In the tests, they compared the performance of routing with partial orders (RPO) with the standard Rapid Spanning Tree Protocol (RSTP) in identical topologies using the same Linux-based software switches. For TCP tests they compared throughput bandwidth, and for UDP tests they compared loss rates.

Focusing on performance evaluation, they deployed the system as a 4x4 torus, with two hosts per switch, in a 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 Inter-Arrival 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 VII and VIII. For TCP, RPO at 0.5sec IAT provides 11.2% better throughput $((3.78Gbps - 3.4Gbps)/3.4Gbps)$ at six times the load $(3sec/0.5sec)$ of RSTP at 3sec IAT. For UDP, RPO at 0.25sec IAT provides roughly the same loss rate and good-put at six times the load of RSTP at 1.5sec IAT.

VIII. CONCLUSION

We have given an overview of routing with partially ordered requirements-based and presented a number of scenarios that demonstrate the power of this paradigm. Specifically, partially ordered QoS and TE requirements enhance network routing to compute a *best set of routes* that satisfy the full range of QoS and TE requirements supported by a given network environment.

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).

The use of partially-ordered requirements 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 forwarded 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 [29]; these results have been verified by an independent testing lab using an untuned *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 software upgrade. Furthermore, implementing these functions using requirements-based routing results in significantly more robust services as they are implemented in the network layer where they have knowledge of the network's topology as it evolves.

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

References

[1] B. R. Smith and P. S. Tatarsky, "Beyond Best Effort: Routing with Requirements," in *The Seventeenth International Conference on Evolving Internet INTERNET 2025*, Mar. 2025.

[2] D. Clark *et al.*, *New Arch: Future Generation Internet Architecture*, Dec. 2003.

[3] S. Floyd and M. Allman, *RFC 5290: Comments on the usefulness of simple best-effort traffic*, Request For Comments, 2008.

[4] V. Cerf and R. E. Kahn, "A Protocol for Packet Network Intercommunication," *Communications, IEEE Transactions on*, vol. 22, no. 5, pp. 637–648, Jan. 1974.

[5] A. Sanchez-Monge and K. G. Szarkowicz, *MPLS in the SDN Era*. Sebastopol, CA: O'Reilly Media, Dec. 2015.

[6] J. J. Garcia-Luna-Aceves, B. R. Smith, and J. T. Samson, "QoS routing using dominant-distance vectors," in *Proceeding IEEE/ACM International Symposium on Quality of Service (IWQoS 2022)*, Jun. 2022.

[7] J. L. Sobrinho and M. A. Ferreira, "From non-optimal routing protocols to routing on multiple optimality criteria," *IEEE/ACM Trans. Netw.*, vol. 31, no. 1, pp. 294–307, Feb. 2023.

[8] L. R. Ford, "Network flow theory," RAND, Tech. Rep. P-923, Aug. 1956.

[9] D. O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus, *RFC 2702: Requirements for traffic engineering over mpls*, Request For Comments, Sep. 2008.

[10] I. Minei and J. Lucek, *MPLS-Enables Applications: Emerging Developments and New Technologies*. Wiley, Jun. 2010.

[11] D. Katz, D. M. Yeung, and K. Kompella, *Traffic Engineering (TE) Extensions to OSPF Version 2*, Request For Comments, Sep. 2003.

[12] 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.

[13] M. Gavanelli, "Partially ordered constraint optimization problems," *Principles and Practice of Constraint Programming — CP Lecture Notes in Computer Science*, vol. 2239, T. Walsh, Ed., pp. 763–768, Nov. 2001.

[14] C. Jongsma, *Introduction to Discrete Mathematics via Logic and Proof*, 1st. Springer, Nov. 2019.

[15] B. R. Smith and J. T. Samson, "Herding Packets: Properties Needed of Metrics for Loop-Free & Best Forwarding Paths," in *Proceedings International Conference on Computing, Networking and Communications (ICNC)*, Jan. 2017.

[16] S. Previdi, C. Filsfils, B. Decraene, M. Horneffer, and R. Shakir, *Source Packet Routing in Networking (SPRING) Problem Statement and Requirements*, IETF Request for Comments, May 2016.

[17] C. Filsfils *et al.*, *Segment Routing Architecture*, IETF Request for Comments, Jul. 2018.

[18] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann, 2000.

[19] *Itu g.114 : One-way transmission time*, ITU-T Rcommendation G.114, 2003.

[20] J. Kindervag, *No more chewy centers: The zero trust model of information security*, Forrester Research Technical Report, Mar. 2016.

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

[22] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co., 1979.

[23] T. J. Schaefer, "The Complexity of Satisfiability Problems," in *10th ACM Symposium on the Theory of Computing*, 1978, pp. 216–226.

[24] S. Garfinkel, J. M. Abowd, and C. Martindale, "Understanding database reconstruction attacks on public data," en, *Commun. ACM*, vol. 62, no. 3, pp. 46–53, Feb. 2019.

[25] A. Nadel and V. Ryvchin, "Efficient SAT solving under assumptions," in *Theory and Applications of Satisfiability Testing – SAT 2012*, ser. Lecture notes in computer science, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 242–255.

[26] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Computer Communication Review*, vol. 38, no. 2, Mar. 2008.

[27] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," en, *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[28] B. Pfaff *et al.*, "The design and implementation of open vSwitch," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, Oakland, CA: USENIX Association, May 2015, pp. 117–130, ISBN: 978-1-931971-218.

[29] 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.