Enabling High Bandwidth Networking as a Service (NaaS)

Greg M. Bernstein and Young Lee, *Member, IEEE*

***Abstract*— when optimizing networked applications that require multiple streams of high bandwidth communications, network bandwidth constraints need to be taken into account. This general problem is referred to as cross-stratum optimization. However, application providers, compute providers (data centers), and network providers may be separate business entities and may want to share a minimal amount of information. In addition, by minimizing the amount of information shared one also promotes interface efficiency particularly when information changes with time. This paper provides an architecture and optimization methods that enable high bandwidth networking as a service by giving new techniques for representing and sharing network bandwidth constraint information. The best representation can be dependent on both the networks data plane, control plane technologies, information hiding requirements, and interface efficiency considerations. We show that depending on the data and control plane that either a list of abstract link path vectors with enumerated shared bottlenecks or an abstract capacitated graph well represents network bandwidth constraints. Taking into account information hiding and interface efficiency we then give methods to further minimize the information transferred in these two approaches with varying degrees of fidelity.**

***Index Terms*—cross stratum optimization, joint network/compute optimization, global load and network balancing.**

I. Introduction

NETWORKED applications, such as those running in, or between multiple data centers and users make use of both compute and network resources. As shown in Figure 1 multiple distinct entities with differing business relationships may be involved in the delivery of a networked application. These can include: (a) application providers, (b) data center providers, (c) network providers, and (d) cross stratum optimizers (aka global load balancers). We are particularly concerned here with the information shared between the network, application provider, and cross stratum optimizer which we show in Figure 1 as the *networking as a service* (NaaS) interface.

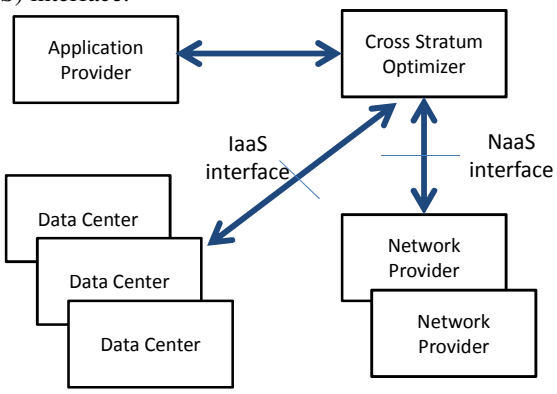


Figure 1. *High level entities and control interfaces associated with cross stratum optimization.*

We use the term *cross stratum optimizer* in preference to that of "load balancer" to emphasize that we are considering cost, constraints, and quality of service in both the network and application strata. We require that this optimization process (a) be dynamic so it can respond to rapidly changing load and resource conditions, (b) require minimal information sharing amongst distinct business entities, (c) be able to deal with a wide range of network data plane and control technologies, and (d) efficiently handle high bandwidth networked applications that utilize multiple concurrent information flows that may vie for limited network resources.

In this paper we show how to best represent and share network bandwidth and cost constraints across the NaaS interface consistent with data plane and control plane limitations and requirements for information hiding and abstraction.

A. Related Work

There is extensive literature in the general area where networks and applications meet, such as on content delivery networks (CDN), web replica placement, etc… Here we review those work that are close to this one in intent and scope.

Early notions of cross stratum optimization can be found in

Manuscript received October 9, 2001. (Write the date on which you submitted your paper for review.) This work was supported in part by the U.S. Department of Commerce under Grant BS123456 (sponsor and financial support acknowledgment goes here). Paper titles should be written in uppercase and lowercase letters, not all uppercase. Avoid writing long formulas with subscripts in the title; short formulas that identify the elements are fine (e.g., "Nd–Fe–B"). Do not write "(Invited)" in the title. Full names of authors are preferred in the author field, but are not required. Put a space between authors' initials.

F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (corresponding author to provide phone: 303-555-5555; fax: 303-555-5555; e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

reference [1] which introduced the notion of dynamic server selection as opposed to static server configuration. In their server selection they consider server load, network bandwidth, and latency as well as client demands in terms of file size. They used measurements of the network to determine bandwidth and latency and didn't assume knowledge of client/server location or network topology.

In reference [2] an efficient measurement based method for deriving approximate network topology was given and applied to consider server selection and application overlay networks. The application overlay networks of [2] are similar to modern multi-flow network applications. A key result of [2] is the quality optimizations possible even with relatively approximate topology information. They do not, however, consider bandwidth constraints.

References [1] and [2] utilized measurements of available network capacity. In networks where traffic policing is used such methods would provide erroneous information, i.e., the current policer limits would be measured rather than available network capacity. Further, the capacity of underlying connection oriented packet switching, or circuit switched networks cannot be measured.

Reference [3] deals with application overlay networks and specifically addresses the "bandwidth provisioning problem". Key to their analysis and results is the consideration of the *price of network bandwidth* in their optimization. They do not however, consider price or availability of server resources. They do not assume limits on the amount of bandwidth available from the network.

Reference [4] introduces the key concept of a third party replica selector (optimizer) whose selection criteria are guided by policies that can include both network and server level information. In addition a decentralized algorithmic approach is given. They allow for dynamics in their costs. However they do not consider network bandwidth constraints. We further note that such a third party optimizer as introduced in [4] could also act as a trusted neutral third party separate from either data center, network, or application provider.

Reference [5] introduces the key idea that networks and applications can and should cooperate by sharing limited information over a simple general interface to allow joint resource optimization. Their network/application interface shares a general notion of "distance" between generalized location identifiers called PIDs. They do not explicitly share bandwidth constraint information across this interface, though they show how to change reported "distance" measures in response to increased link loading. Many of the ideas from reference [5] have been incorporated into a relatively recent standards efforts at the IETF on application layer traffic optimization (ALTO)[6].

B. *Contributions and Outline*

This paper is concerned with the NaaS interface for applications that consume significant amounts of bandwidth relative to network capacity. In this case, network bandwidth constraints need to be explicitly communicated between a network and the cross stratum optimizer. We show how to optimally convey network bandwidth constraints for a wide range of networking technologies. Such techniques are useful between large groups of users and data centers, between data centers across a WAN or MAN, within data centers across a LAN, or even within compute clusters.

We start with a discussion of two generic cross stratum optimization problems: one related to server selection, and another related to remote backup scheduling. We show how these can be formulated as linear programming (LP) problems, or mixed integer linear programming (MILP) problems with the network constraints and costs represented as an "augmenting" multi-commodity flow problem. We furnish two versions of this augmenting multi-commodity flow problem based on given paths or a network graph. We then use these formulations along with insight into networks topology to derive exact and approximate minimal representations of the networks bandwidth constraints appropriate for a given network control and data plane technology. Finally, we provide an overview of an interface that can enable high bandwidth NaaS for a wide variety of network data and control planes.

II. Cross Stratum Optimization Problems

In the following, we give fairly general formulations of two cross stratum optimization problems. The first is a variant of a server selection problem that could represent video on demand (VoD) services. The second involves the scheduling of large data transfers such as backups. To deal with different types of network constraints with respect to choices in path we look at these optimization problems as "coupled" to multi-commodity flow problems involving the network resources. We then quickly review two different multi-commodity flow formulations that are useful for the different path choice restrictions that we derived in section IV.

A. *Multiple Server Selection/Video On Demand Problem*

In Figure 2, we show a number of data centers used to serve clients located in a number of end user regions. For concreteness, suppose that the data centers are serving video on demand (VoD) streaming services to client regions.

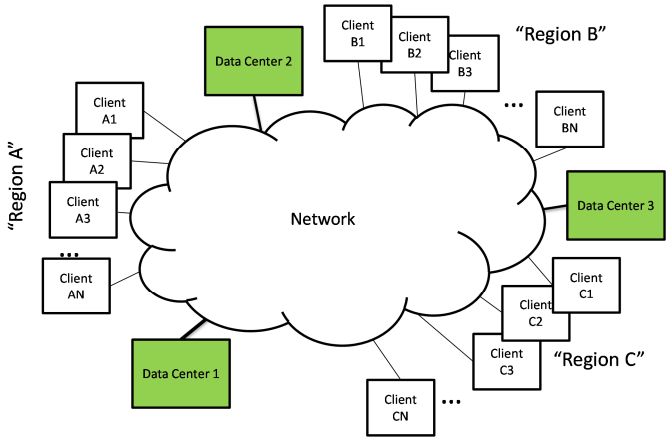


Figure 2. *Server selection/VoD scenario.*

Suppose we had S data centers, DC_s, and that each can stream a maximum quantity Q_s of video. We have R users regions, U_r, that have demands D_r for video that must be satisfied. Each DC_s has a cost of cd_s to stream a unit of video Q. In addition, the communication of video from DC_s to U_r across the network is subject to network costs and capacity constraints. Let q_{sr} be the total bandwidth of video produced at DC_s for U_r. The objective is to minimize the total cost (data center and network):

$$\sum_s cd_s q_s + \sum_{s,r} networkCost(q_{sr}) \quad (1.1)$$

Subject to the network capacity constraints as well as:
Data center capacity constraints:

$$\sum_r q_{sr} \le Q_s \quad \forall s \quad (1.2)$$

And user demands:

$$\sum_s q_{sr} = d_r \quad \forall r \quad (1.3)$$

B. *Scheduled Remote Backups*

In Figure 3, we show multiple data centers connected to a network. In this scenario, data center DC_s needs to do a bulk transfer of data to one or more remote data centers, DC_r, within prescribed time limits. In particular there is Q_s bytes of data to be transferred between DC_s and the set of possible remote data centers $\mathbf{DC}_r(s)$, and the data transfer must start by $Start_s$ time and end before End_s time.

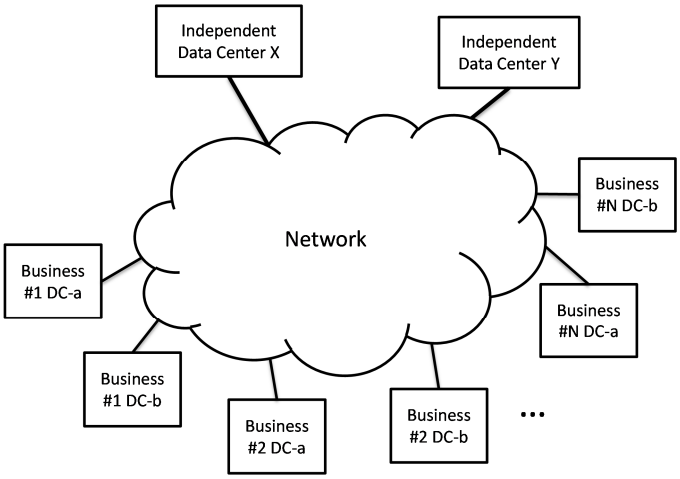


Figure 3. *Data Center scheduled data transfer scenario.*

Let $rate_t^{sr}$ denote the rate that DC_s is transferring data to DC_r during time period $t = 0,1,...,T$. Here we break down the time interval over which scheduling is to occur, e.g., 24 hours, into a fixed number of equally spaced periods of length τ in time. The requirement that all the data must be transferred can be expressed as

$$\sum_{r \in \mathbf{DC_r}(s)} \tau \sum_t rate_t^{sr} = Q_s \quad \forall s \quad (1.4)$$

Suppose that each remote data center imposes a storage cost of $storageCost_r$ regardless of when the transfer starts or stops, then a simple linear objective function could be:

$$\alpha \sum_r storageCost_r \tau \sum_{s,t} rate_t^{sr} + \sum_{sr,t} networkCost(rate_t^{sr}, t) \quad (1.5)$$

Where the parameter $\alpha \ge 0$ in equation (1.5) can be used to emphasize or de-emphasize the relation of storage costs to networking costs. A more realistic objective function could include a sum of (non-linear) monotonically decreasing functions of the transfer rates to encourage faster transfers as suggested in [7].

C. *Multi-Commodity Flow with Path Selection*

When the technology permits arbitrary path selection we can formulate network costs and constraints via a multi-commodity flow problem [7]. This formulation results in a linear programming problem which can be solved very efficiently. However, such a formulation allows the demand between a source and destination node to be split amongst multiple paths. The "conformal realization theorem" of [7] guarantees that we could decompose such an optimal flow solution into a set of fixed resource paths. Hence if appropriate inverse multiplexing like technology is available one can directly make use of the solution. Otherwise we need to restrict "splitting" by making use of integer valued (or binary) flow variables, this leads to a mixed integer linear program (MILP) which are theoretically difficult to solve, but amenable to approximate solution techniques.

Let $G(N, L)$ be the graph of a network with n nodes and m links. Let x_{ij}^{sr} denote the amount of bandwidth used from source, s, for receiver, r, on link (i, j). Let q_{sr} be the total

demand between *s* and *r*, cn_{ij} the costs per link, and bw_{ij} the capacity of link (i, j).

The network cost function can be expressed as:

$$\sum_{s,r}\sum_{(i,j)} cn_{ij} x_{ij}^{sr} \tag{1.6}$$

Link bandwidth constraints:

$$\sum_{s,r} x_{ij}^{sr} \le b_{ij} \quad \forall (i,j) \tag{1.7}$$

Conservation of flow constraints at each node:

$$\sum_{\{j|(i,j)\in L\}} x_{ij}^{sr} - \sum_{\{j|(j,i)\in L\}} x_{ji}^{sr} = \begin{cases} q_{sr} & \text{if } i = s \\ -q_{sr} & \text{if } i = r \\ 0 & \text{otherwise} \end{cases} \tag{1.8}$$

Where the equation holds $\forall i, s, r$.

In the solution of these optimization problems, the non-zero link flow variables, x_{ij}^{sr} ,determine the communications paths through the network. If we wish to restrict the paths for flows between source, *s*, and receiver, *r*, from using link (i, j) we can set $x_{ij}^{sr} = 0$. Similarly if we wish to restrict the choice to a specific path, *p*, through the network we can set to a single route we can set $x_{ij}^{sr} = 1 \,\forall (i,j) \in p$ and $x_{ij}^{sr} = 0 \,\forall (i,j) \notin p$.

D. Multi-Commodity Flow with Given Path Choices

The case of a reasonable number (>1) of fixed choices for the path, *p,* between a source and destination cannot be easily incorporated into the previous formulation. In such a case the multi-commodity flow formulation suggested in [8] for efficient solution to the routing and wavelength assignment (RWA) problem in WDM networks is useful. Let $\mathbf{W}$ denote the set of source-receiver (s, r) pairs to be considered. Let $\mathbf{P}_{sr}$ denote the set of paths that may be used for the source-destination pair (s, r) . Let q_{sr} denote the bandwidth demand of the source-destination pair, and $y_{p,sr} = 0,1$ indicate whether a particular path *p* between the source-destination pair is used. Note that we can allow $y_{p,sr} \in [0,1]$ if inverse multiplexing mechanisms are available. The link bandwidth constraint can be formulated as:

$$\sum_{(s,r)\in \mathbf{W}} \sum_{\{p|(i,j)\in p\}} q_{sr} y_{p,sr} \le b_{ij} \tag{1.9}$$

The demand constraint can be formulated as

$$\sum_{p\in \mathbf{P}_{sr}} y_{p,sr} = 1 \quad \forall sr \tag{1.10}$$

Note that links may be given costs or entire paths may be given costs to further hide link details.

Network cost:

$$\sum_{(s,r)\in \mathbf{W}} \sum_{p\in \mathbf{P}_{sr}} \sum_{\{(i,j)|(i,j)\in p\}} cn_{ij} q_{sr} y_{p,sr} \tag{1.11}$$

Or we can express this in terms of cost per path

$$\sum_{(s,r)\in \mathbf{W}} \sum_{p\in \mathbf{P}_{sr}} \text{cost}(p) q_{sr} y_{p,sr} \tag{1.12}$$

III. Information Hiding and Interface Efficiency

The previously mentioned cross stratum optimization problem formulations made use of network topology, link cost, and link capacity constraint information, as well as data center cost and capacity information. In practical networks, such information can be quite voluminous and generally considered proprietary by application, network, and data center providers. In this section, we look at the minimal essential information to be conveyed between the network, and the cross stratum optimizer, i.e., across the NaaS interface as depicted in Figure 1. We break this discussion into two parts. First, we address the general case where the network would provide a list of paths and one wishes to minimize the corresponding accompanying constraint information. Second, we consider graph representations and how these may be simplified. Since application demand information may be sensitive to the application provider, we consider both the cases where no demand information is conveyed to the network, and where bounds on application demands are given. We always assume that possible source and receiver pairs are made available to the network and it is from this information that the network will provide abstracted path list or graph information.

A. Path List Bandwidth Constraint Reduction

If a limited selection of paths are provided by the network to the application optimizer, then by providing costs per path as in equation (1.12) no information on individual link costs need to be given. Looking at the link capacity constraint equations (1.9) it initially appears that we need to convey information on every link in the network across the NaaS interface since the application demands q_{sr} are not given to the network. However, if we know a constraint equation for a particular link will always be satisfied then that link information does not need to be sent to the optimizer furthering information hiding and interface efficiency. For example, for a link (i, j) that is not used by any path the left hand side of equation (1.9) is zero and hence the constraint is always satisfied and there is no need to reveal the value of b_{ij}, the capacity of link (i, j). In Figure 4, we show a network with five given paths and thirteen links. Links L3 and L10 are not used by any of the paths and hence their bandwidth constraints are always satisfied. For the next more complicated case consider path P3 in Figure 4. P3 is the only path given that uses links L7, L11, and L13 and doesn't share any links with any other path. The network does not need to specify the link capacities for each of these links but only the smallest (bottleneck) capacity link since equation (1.9) will be satisfied for the larger capacity links along path P3. In this case the network can report the capacity of the path as the minimum of the links along the path, i.e., $cap(path) = \min_{link\in path}(bw_{link})$ and doesn't need to reveal which particular link forms the potential bottleneck or how many links are involved in the path.

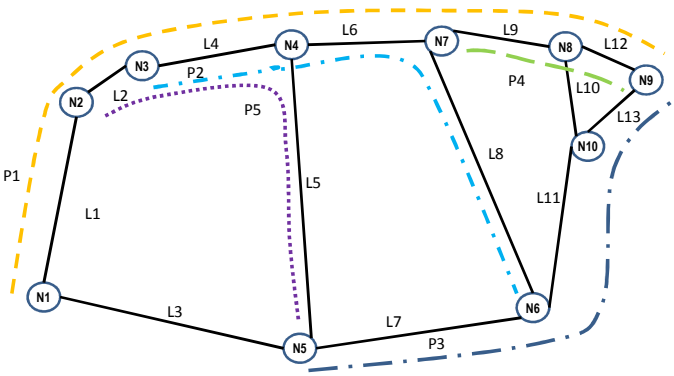


Figure 4. *Example network with links constraints and given paths.*

In general *we can remove a constraint equation for a link if there is another link carrying the same set (or more) of paths that has less capacity, i.e., another link would provide a tighter bound.* This rule leads to the following algorithm.

Algorithm 1:

(A1) For each link determine the subset of all given paths that traverse it. Call these subsets *path sets*.

(A2) Group links that have exactly the same set of paths traversing them. Call these links the *contained links* for the *path set*.

(A3) Find the minimum capacity link over all links in a path set. Call this the *path set minimum*

(A4) If a path set is wholly contained in another path set and the minimum of the containing path set is smaller than the contained path set then the bandwidth constraint of the contained path set is always satisfied and can be dropped.

(A5) Each path set not eliminated in the previous step presents a mutual bandwidth constraint over the paths in the set and can be represented by an "abstract" link with capacity corresponding to the path set minimum. This abstracted information is then exchanged between network stratum and application stratum.

1) Example

Suppose the links in Figure 4 have the following capacity: B1 = 10, B2=7, B3=5, B4=6, B5=9, B6=12, B7=8, B8=11, B9=5, B10=7, B11=8, B12 = 7, B13=5. In Table 1, we show the result of applying steps (A1)-(A3) of Algorithm 1.

Table 1. *Path sets, links, and path set minimums.*

Path sets	Contained Links	Path Set Minimum
P1	L1	10
P2	L8	11
P3	L7, L11, L13	5 (abstract link A)
P5	L5	9
{P1, P2}	L6	12
{P1, P4}	L9, L12	5 (abstract link B)
{P1, P5}	L2	7
{P1, P2, P5}	L4	6 (abstract link C)

For step (A4) of Algorithm 1, we look at the containment relationships between the path sets and their capacities. In particular, path set {P1} is contained in path set {P1, P2, P5} and min(P1) > min({P1, P2, P5}) so link L1's constraint can be dropped. Set {P2} is contained in {P1, P2, P5} and min(P2) > min({P1, P2, P5}) so link L8's constraint can be dropped. Set {P3} is not contained in any other set so its constraint must be kept. Set {P5} is contained in {P1, P2, P5} and min(P5) > min({P1, P2, P5}) so link L5's constraint can be dropped. Set {P1, P2} is contained in {P1, P2, P5} and min({P1,P2}) > min({P1, P2, P5}) so path set {P1,P2} corresponding link L6's constraint can be dropped. Set {P1,P4} is not contained in any other set so its constraint must be kept. {P1, P5} in {P1, P2, P5} and min({P1,P5}) > min({P1, P2, P5}) so L2's constraint can be dropped. Finally, set {P1, P2, P5} is not contained in any other set so its constraint must be kept.

Applying step (A5) we end up with three abstract links (A, B, C) with capacity (5, 5, 6) and the following sharing:
P3 uses abstract link A (not shared), P1 uses abstract link B and C, P2 uses abstract link C, P4 uses abstract link B, P5 uses abstract link C. Note how potentially thirteen link capacity constraints have been reduced to the following three constraint equations, a significant information transfer savings and information hiding.

$$\begin{aligned} q_{N5N9}\, y_{P3,N5N9} &\le 5 \\ q_{N1N9}\, y_{P1,N1N9} + q_{N7N9}\, y_{P4,N7N9} &\le 5 \\ q_{N1N9}\, y_{P1,N1N9} + q_{N3N5}\, y_{P2,N3N6} + q_{N2N5}\, y_{P5,N2N5} &\le 6 \end{aligned} \qquad (1.13)$$

Where q_{NsNr} are the demands between nodes *Ns* and *Nr* and $y_{Pi,NsNr}$ are the path flow variables. Note that since the demands are unknown to the network stratum, we cannot reduce the set of equations (1.13) any further.

B. Path List Constraint Reduction, Known Demands

Now suppose that bounds $Q_{sr} \ge q_{sr}$ on q_{sr} ,the bandwidth demand of the source-destination pair, are given by the application stratum to the network stratum across the NaaS interface. We can use the results of Algorithm 1 together with these bounds to possible reduce further the number of constraint equations. From $y_{p,sr} = 0,1$ or $y_{p,sr} \in [0,1]$ and equation (1.10) we obtain the following upper bound for the left hand side of equation (1.9):

$$\sum_{(s,r)\in \mathbf{W}} \sum_{\{p|(i,j)\in p\}} q_{sr}\, y_{p,sr} \le \sum_{\substack{(s,r)\in \mathbf{W} | \exists P_n \\ \text{from s to r} \\ \text{traversing } (i,j)}} Q_{sr} \quad \forall (i,j) \qquad (1.14)$$

Where the right hand side of (1.14) is just the sum of all demands that have a path that traverses link *(i, j)*. If the right hand side of (1.14) is also less than b_{ij} then the capacity constraint for link *(i, j)* is always met.

Algorithm 2:

(A1) Use Algorithm 1 to compute the paths sets that represent potential bandwidth bottlenecks.

(A2) For each path set sum the demand bounds for each demand that has at least one corresponding path.

(A3) If the sum of the corresponding demand bounds is less than that the path set minimum, then the constraint corresponding to this path set is always met and does not need to be conveyed to the optimizer.

1) Example with Demand bounds

In Figure 5 we show our example network with an enhanced group of paths that includes overlapping paths between same source and destination. In Table 2 we show the results of applying algorithm 1.

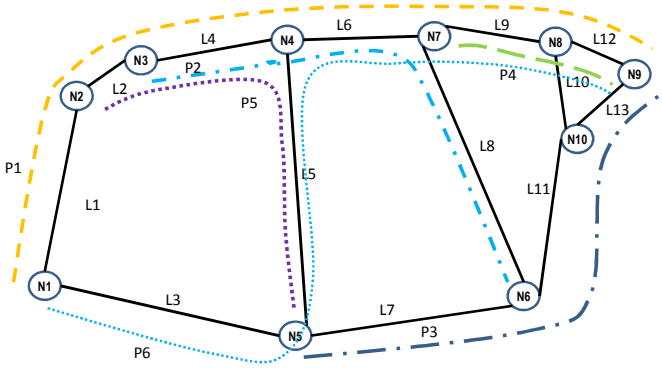


Figure 5. *Example network with links constraints and multiple given paths between N1 and N9.*

Table 2. *Path sets, links, and path set minimums.*

Path sets	Contained Links	Path Set Minimum
P1	L1	10
P2	L8	11
P3	L7, L11, L13	5 (abstract link A)
P6	L3	5
{P1, P5}	L2	7
{P1, P2, P5}	L4	6 (abstract link B)
{P1, P2, P6}	L6	12 (abstract link C)
{P5, P6}	L5	9 (abstract link D)
{P1, P4, P6}	L9, L12	5 (abstract link E)

Before taking into account the demand bounds, we get the following set of constraint equations induced by the constraining path sets of *Algorithm 1*(highlighted in red in Table 2):

$$\begin{aligned} q_{N5N9}\,y_{P3,N5N9} &\le 5 \\ q_{N1N9}\,y_{P1,N1N9} + q_{N3N5}\,y_{P2,N3N6} + q_{N2N5}\,y_{P5,N2N5} &\le 6 \\ q_{N1N9}\,y_{P1,N1N9} + q_{N3N5}\,y_{P2,N3N6} + q_{N1N9}\,y_{P6,N1N9} &\le 12 \\ q_{N2N5}\,y_{P5,N2N5} + q_{N1N9}\,y_{P6,N1N9} &\le 9 \\ q_{N1N9}\,y_{P1,N1N9} + q_{N7N9}\,y_{P4,N7N9} + q_{N1N9}\,y_{P6,N1N9} &\le 5 \end{aligned} \quad (1.15)$$

Note that the path sets {P1, P2, P6} and {P1, P4, P6} both contain paths P1 and P6 that serve the same source-destination pair, i.e., could be used to satisfy the demand between nodes N1 and N9. Using the bounds furnished by equation (1.14) gives:

$$\begin{aligned} q_{N5N9}\,y_{P3,N5N9} &\le Q_{N5N9} \\ q_{N1N9}\,y_{P1,N1N9} + q_{N3N6}\,y_{P2,N3N6} + q_{N2N5}\,y_{P5,N2N5} &\le Q_{N1N9} + Q_{N3N6} + Q_{N2N5} \\ q_{N1N9}\,y_{P1,N1N9} + q_{N3N6}\,y_{P2,N3N6} + q_{N1N9}\,y_{P6,N1N9} &\le Q_{N1N9} + Q_{N3N6} \\ q_{N2N5}\,y_{P5,N2N5} + q_{N1N9}\,y_{P6,N1N9} &\le Q_{N2N5} + Q_{N1N9} \\ q_{N1N9}\,y_{P1,N1N9} + q_{N7N9}\,y_{P4,N7N9} + q_{N1N9}\,y_{P6,N1N9} &\le Q_{N1N9} + Q_{N7N9} \end{aligned} \quad (1.16)$$

Suppose we have the following demands:

Table 3. *Source Destination demand bounds and Paths.*

Source Destination	Demand bound	Paths
N1, N9	5	P1, P6
N2, N5	6	P5
N3, N6	2	P2
N5, N9	4	P3
N7, N9	4	P4

Then

$$\begin{aligned} Q_{N5N9} &= 4 \quad \{P_3\} \\ Q_{N1N9} + Q_{N3N6} + Q_{N2N5} &= 13 \quad \{P_1, P_2, P_5\} \\ Q_{N1N9} + Q_{N3N6} &= 7 \quad \{P_1, P_2, P_6\} \\ Q_{N2N5} + Q_{N1N9} &= 11 \quad \{P_5, P_6\} \\ Q_{N1N9} + Q_{N7N9} &= 9 \quad \{P_1, P_4, P_6\} \end{aligned} \quad (1.17)$$

Comparing equations (1.15)-(1.17) we see that the bounds induced by the demand constraints for path sets $\{P_3\}$ and $\{P_1, P_2, P_6\}$ are tighter than the link constraints for these path sets and hence will always be satisfied and do not need to be communicated from the network stratum to the application stratum reducing the number of constraints to be shared from five to three.

C. Service Specific Graph Reductions

We have previously noted that supplying long lists of potential paths may be less interface efficient than supplying some type of graph representation of the network. Here we look at the case where the network chooses to supply a cost-constraint graph to the optimizer. The network provider's internal network view may be quite complex consisting of multiple networking layers such as WDM, TDM, MPLS, Ethernet, or IP. On the other hand, most applications will be concerned with the combined impact on the single layer that they interface to the network. In addition, the set of potentially communicating nodes of interest to the application would be limited and the possible communicating pairs would be explicitly conveyed across the NaaS by the application to the network. The network would then provide an appropriate simplified (reduced) graph representation to the optimizer. Initially this problem sounds similar to the *topology aggregation* (TA) problem in hierarchical routing systems [9]. In TA one seeks to abstract the internals of a subdomain from the point of view of its border nodes for use in hierarchical routing. In this sense, TA is similar to a provider coming up with an abstract graph to represent network resources available to potential source-destination pairs. However, current TA techniques work by simplifying full mesh representations between border nodes and thus begin by hiding the bandwidth constraint dependencies between potential paths that are critical for the high bandwidth case. Note that when one uses TA in routing, subsequent path setups will trigger a "re-aggregation" if significant changes to resource availability levels have occurred.

1) Generating Service Specific Graphs from Paths

Although TA is known to be a difficult problem, and it would appear that our need to retain bandwidth constraint information makes it even more difficult. From the practical point of view, the limited number of possible communicating nodes allows a very reasonable solution as follows. In addition to the application furnishing the set of candidate

communicating nodes across the NaaS interface, it would also indicate in either an exact or approximate sense its optimization goals, e.g., lowest latency, lowest cost, high reliability, etc… The network can then generate a number of candidate paths, not to send across the NaaS interface, but to derive a service specific graph that retains only those links and nodes that are utilized by the candidate paths. The resulting graph is then sent across the NaaS interface rather than a very long list of candidate paths and their mutual constraints. Such an approach is practical due to the existence of very efficient algorithms for finding k-shortest paths [10].

As an example consider the network shown in Figure 6 where the candidate communicating entities are: ("A4","B9"), ("A14", "B4"), ("B28","E5"), and ("A17","D4").

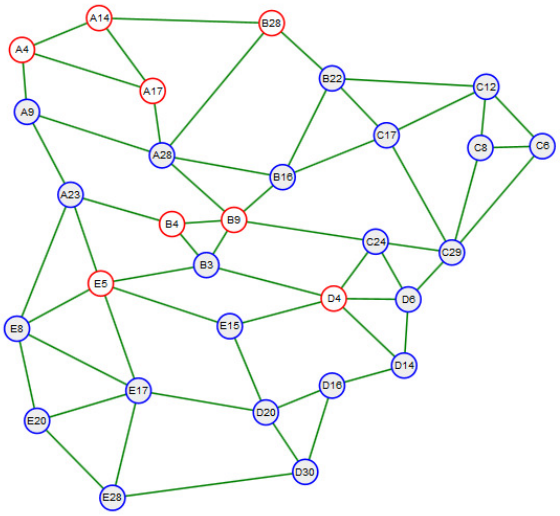


Figure 6. *Network to be reduced.*

In Figure 7 we show service specific graphs derived from the graph of Figure 6. In Figure 7(a) we show a service specific graph optimized for lowest latency derived from the k-shortest paths between the candidate communicating nodes based distance as a weight (k=4). In Figure 7(b) we show a graph optimized for lowest cost derived from the k-shortest paths (k=4) based on costs (randomly assigned). While in Figure 7(c) we show a graph optimized for low latency and high reliability derived from the k-shortest disjoint pairs of paths [11] (k=4).

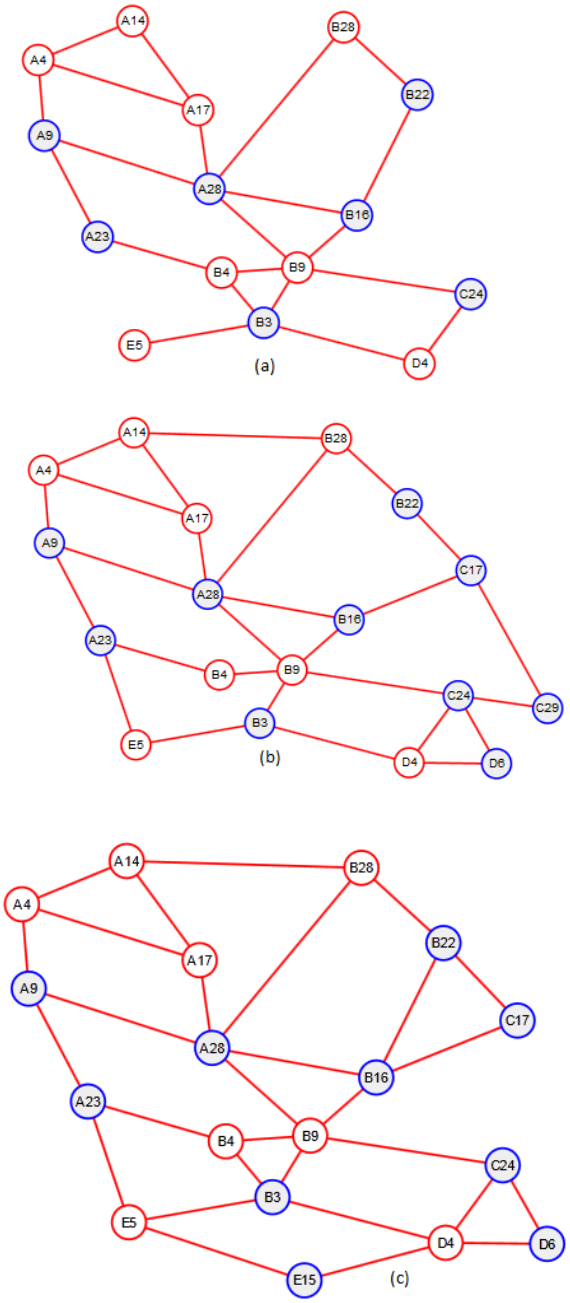


Figure 7. *Service specific graphs. (a) lowest latency, (b) lowest cost, (b) low latency and high reliability.*

2) Reducing Service Specific Graphs

Given a service specific network graph with costs and capacities generated in any manner we can further reduce the information to be transferred across the NaaS via any of the following rules.

(C1) "Stub" nodes, i.e., nodes of degree 1, not in the source-destination set along with their accompanying links can be removed.

(C2) **Nodes of degree 2**, i.e., nodes forming a linear segment, not in the source-destination set, can be replaced by a link whose cost is the sum of the costs

of the two incident links and whose bandwidth is the minimum.. .

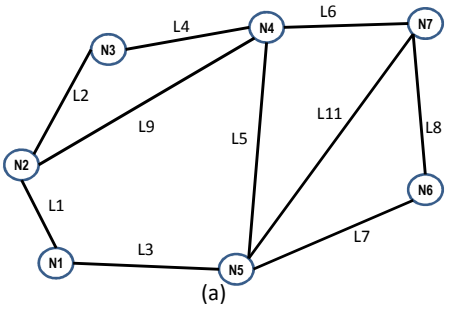
(C3) ***Hiding high capacity links***. Look at both nodes bounding a link of relatively high capacity compared to others in the network. Look at all the incident edges at each node other than the edge of interest. If the sum of these capacities is less than that of the edge of interest then the nodes can be merged. Note however the accuracy of the cost for traversing this edge would be lost. This procedure can be run on all links attached to non-SD nodes. Note that we can try to abate some of the "cost distortion" by distributing some of the cost of the removed link to the links adjacent to the newly merged node. If we allow for some "capacity distortion", i.e., our high capacity link's capacity isn't strictly greater than the sum of links adjacent to end nodes we could possibly reduce the capacity of the links adjacent to the end nodes.

(C4) ***Hiding high capacity subnets***. By repeated application of rule (C3) one can reduce a high capacity subnet to a single node.

Although rules (C1) and (C2) may appear somewhat trivial they are very powerful when applied to ring sub-networks commonly seen in optical networks and to tree sub-networks commonly seen in Ethernet networks.

3) Example with Multiple Spanning Tree Ethernet

In Figure 8(a) we show an Ethernet network. This network supports multiple spanning tree protocol (MSTP). The application is interested in communications between nodes N1, N3, N5, N6, and N7. In Figure 8(b)-(d) we show a multiple spanning tree instances (MSTI) along with the reduced graph obtained via rules (C1) and (C2). For tree graphs rules (C1) and (C2) always allow us to remove all nodes other than those of interest. In Figure 8(b)-(d) we denote by "Lx m Ly" the abstracted link whose cost is the sum of the costs of Lx and Ly and whose capacity is the minimum of their capacities. Note how the bandwidth implications of the different spanning trees become more apparent in the reduced representations.



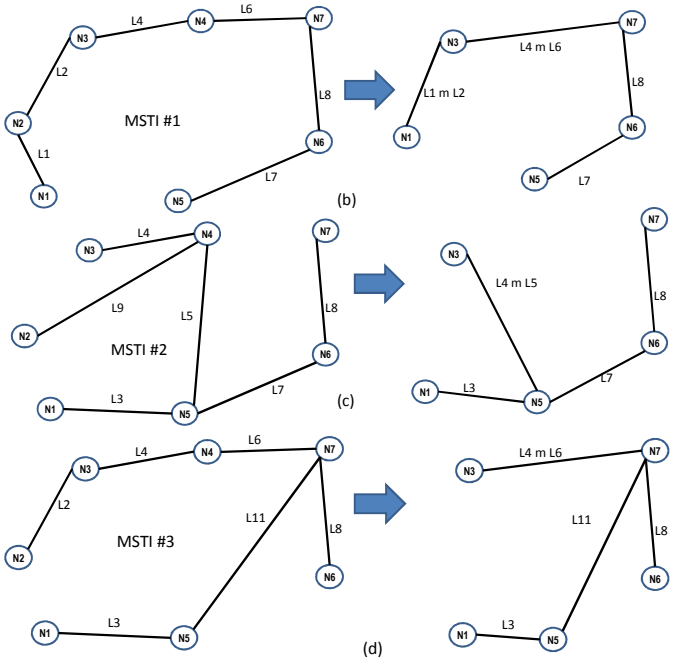


Figure 8. *Ethernet network (a), along with multiple spanning tree instances and their reductions (b)-(d).*

D. Service Specific Graph Reductions with Known Demands

Suppose that bounds on the source-destination demands are given, i.e., $Q_{sr} \geq q_{sr}$ for all q_{sr}. Under the mild assumption that no chosen path contains a loop, i.e., the paths are simple, we can then bound the traffic on any link by the sum of the demands. In particular

$$\sum_{s,r} x_{ij}^{sr} \leq \sum_{s,r} Q_{sr} \quad \forall (i,j) \tag{1.18}$$

Since each $x_{ij}^{sr} \leq q_{sr} \leq Q_{sr}$. Now if for some link *(i, j)* we have

$$\sum_{s,r} Q_{sr} \leq b_{ij} \tag{1.19}$$

Then the bandwidth constraint for that link (equation (1.7)) will always be satisfied. This leads to the following possible graph reduction rule:

(**D1**) For any link satisfying equation (1.19) one can merge the endpoint nodes if they are not source or destination nodes for any flow.

As mentioned in the previous section such node mergers can cause "distortion" in the costs of computed paths .

1) Example Transformation with Demands

Consider the network graph given in Figure 4. Suppose we are interested in the source/destination nodes N1, N3, N5, N6, N7, N9, and N10. Applying rule (C2) at node N2 gives the graph shown in Figure 9(a). Now if links L4 and L9 are high capacity links meeting the criteria of rule (D1) then we can "merge" node N4 into node N3, and node N8 into node N7 to obtain the reduced graph shown in Figure 9(b).

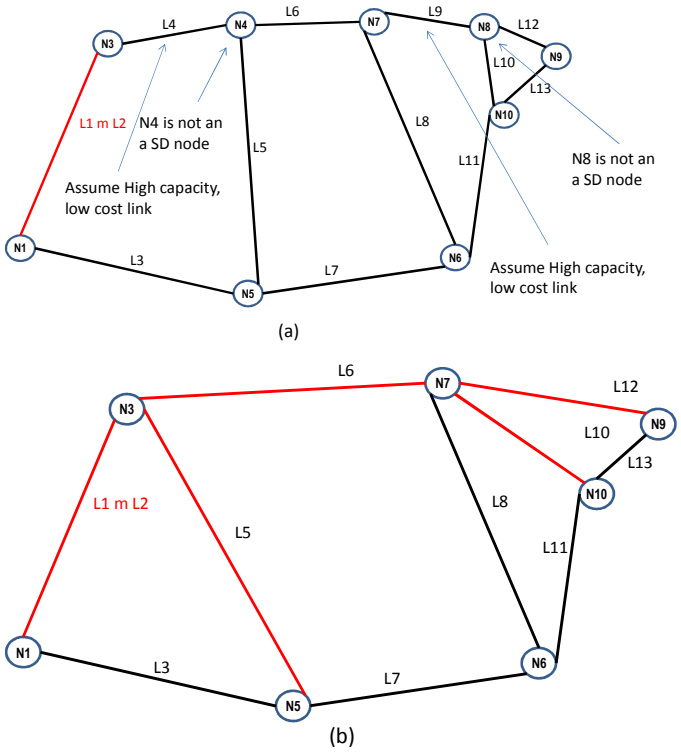


Figure 9. *Approximate graph transformations.(a) Original network, (b) transformed network.*

IV. The Networking as a Service Interface

The job of the cross stratum optimizer from the network perspective is to choose suitable paths for use by the application as part of its overall optimization. The NaaS interface will furnish information from the network on which to make this choice. We have shown that there are two basic ways to convey this information (a) via lists of path along with their corresponding cost, capacity, and mutual constraint information, or (b) via one or more cost-constraint graphs. Given a graph one can derive paths hence a list of paths is in some sense more fundamental representation. In addition lists of paths may provide the highest degree of information hiding between the network and application stratum. However, in server selection problems or other network applications where there may be many potential communicating entities a graph representation may be much more efficient than very long lists of all potential paths even in the case where there is only one path choice per source and destination pair. The data plane, control plane, and network provider preferences all influence whether a graph representation can be used in addition to a path list representation.

When arbitrary path placement between potential source and destination nodes is allowed, a graph representation permits full utilization of network resources. Technologies that allow arbitrary placement of paths across a network include: circuit switched technologies (WDM, TDM), strictly connection oriented packet technologies (MPLS, ATM, and Frame Relay), and connection oriented modes of multi-purpose protocols such as InfiniBand's CO service[13]. OpenFlow [14] capable switches permit general forwarding behavior based on general packet header matching. These can include Ethernet destination and source addresses, IP destination and source addresses, as well as other protocol related fields. Since both source and destination information can be utilized in forwarding OpenFlow can enable traffic engineering like a connection oriented packet switching technology.

In other technologies there is only a single path choice between potential source and destination nodes. For example in IP, a connectionless technology, one typically thinks of a single path between each source and destination (not considering equal cost multipath). Here there are two discernible cases: (a) the paths are derivable from graph, (b) the paths are not derivable from graph (or the provider chooses not share a graph). For example in the case of single area OSPF the paths can be derived from a graph, while BGP [15] uses techniques based on policies and path vectors (AS_PATH) as part of its route selection process and these are not derived from graphs. Another important example where the paths are easily derived from a graph comes from the basic Ethernet Bridge specifications (802.1D) [16] utilizes a single tree structure (graph) as the communication backbone between all nodes.

Finally, there are a number of interesting technologies where more than one path choice between source and destination is available but arbitrary selection is not available or permitted. Multi-Topology routing enhancements to OSPF [16] basically furnishes multiple graphs with the shortest path used on each graph. This could be represented by multiple graphs with the constraint that only one choice (the shortest path) is used from each graph. IEEE 802.1Q [18] includes virtual LANs (VLANs) and allows for multiple spanning trees each of which can be considered a graph. The multiple spanning tree protocol (MSTP) allows for the assignment of VLANs to different spanning trees. Hence we have more than one choice in paths but all flows within the same VLAN have to share the same tree. This could be represented by multiple graphs with the constraint that all paths must be selected from the same graph.
In some cases, for example, in WDM networks due to optical impairments, the usable paths may be restricted in a way not readily discerned from a simple graph representation.

Table 4 summarizes where graph representations may be appropriate.

Table 4. *Technologies, paths, and graphs.*

Between Source and Destination	**Representation**	**Example Technologies**
Arbitrary Potential Paths	Graph	TDM, WDM, MPLS, OpenFlow
Single Potential Path	Derivable from Graph	IP: OSPF
	Path List (only)	IP: BGP
	Multiple graphs, one	Multi-Topology

Multiple Potential Paths	path choice per graph	OSPF
	One graph choice per multiple trees (graphs)	VLAN assignment in MSTP
	Path List (only)	WDM with impairments

V. Conclusion

In this paper we have shown that high bandwidth cross stratum optimization, a general form of load balancing across both application and network stratum, is enabled by an efficient network as a service interface. Key to this interface is the sharing of bandwidth constraint information that preserves the privacy of both network and application providers. We showed that this may be accomplished either via lists of paths with their mutual constraints reduced via our path-set algorithm or by the generation and reduction of service specific graphs.

Both approaches can readily be incorporated into emerging standards for network-application interaction such as ALTO [6]. In addition, as our examples showed, such techniques have wide applicability from within the data center to between data centers across wide area networks.

References

[1] R. L. Carter and M. E. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *INFOCOM '97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, 1997, vol. 3, pp. 1014–1021 vol.3.

[2] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2002, vol. 3, pp. 1190–1199 vol.3.

[3] Z. Duan, Z.-L. Zhang, and Y. T. Hou, "Service overlay networks: SLAs, QoS, and bandwidth provisioning," *IEEE/ACM Trans. Netw.*, vol. 11, no. 6, pp. 870–883, 2003.

[4] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, "DONAR: decentralized server selection for cloud services," in *Proceedings of the ACM SIGCOMM 2010 conference*, New York, NY, USA, 2010, pp. 231–242.

[5] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: provider portal for applications," in *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*, New York, NY, USA, 2008, pp. 351–362.

[6] R. Alimi, Y. Yang, and R. Penno, "ALTO Protocol." [Online]. Available: http://tools.ietf.org/html/draft-ietf-alto-protocol-14.

[7] D. P. Bertsekas and D. P. Bertsekas, *Network Optimization: Continuous and Discrete Models*. Athena Scientific, 1998.

[8] Asuman E. Ozdaglar and Dimitri P. Bertsekas, "Routing and wavelength assignment in optical networks," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 259 –272, 2003.

[9] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, "Analysis of Topology Aggregation techniques for QoS routing," *ACM Comput. Surv.*, vol. 39, Sep. 2007.

[10] J. Hershberger, M. Maxel, and S. Suri, "Finding the *k* shortest simple paths: A new algorithm and its implementation," *ACM Trans. Algorithms*, vol. 3, no. 4, p. 45, 2007.

[11] Q. V. Phung, D. Habibi, H. N. Nguyen, and K. Lo, "K Pairs of Disjoint Paths Algorithm for Protection in WDM Optical Networks," in *2005 Asia-Pacific Conference on Communications*, 2005, pp. 183 –187.

[12] M. Hillenbrand, V. Mauch, J. Stoess, K. Miller, and F. Bellosa, "Virtual InfiniBand clusters for HPC clouds," in *Proceedings of the 2nd International Workshop on Cloud Computing Platforms*, New York, NY, USA, 2012, pp. 9:1–9:6.

[13] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[14] S. Hares, Y. Rekhter, and T. Li, "A Border Gateway Protocol 4 (BGP-4)," Jan-2006. [Online]. Available: http://tools.ietf.org/html/rfc4271.

[15] "802.1D-2004," 2004.

[16] L. Nguyen, P. Psenak, S. Mirtorabi, P. Pillay-Esnault, and A. Roy, "Multi-Topology (MT) Routing in OSPF." [Online]. Available: http://tools.ietf.org/html/rfc4915.

[17] "IEEE Standard for Local and metropolitan area networks–Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks," *IEEE Std 802.1Q-2011 (Revision of IEEE Std 802.1Q-2005)*, pp. 1 –1365, 2011.

Greg M Bernstein (M '82)
Bio to be provided.
Young Lee
Bio to be provided.
.