Report Number: WUCSE-2006-35

2006-01-01

Efficient Mapping of Virtual Networks onto a Shared Substrate

Jing Lu and Jonathan Turner

Virtualization has been proposed as a vehicle for overcoming the growing problem of internet ossification [1]. This paper studies the problem of mapping diverse virtual networks onto a common physical substrate. In particular, we develop a method for mapping a virtual network onto a substrate network in a cost-efficient way, while allocating sufficient capacity to virtual network links to ensure that the virtual network can handle any traffic pattern allowed by a general set of traffic constraints. Our approach attempts to find the best topology in a family of backbone-star topologies, in which a subset of nodes constitute the... Read complete abstract on page 2.

Efficient Mapping of Virtual Networks onto a Shared Substrate

Jing Lu and Jonathan Turner

Complete Abstract:

Virtualization has been proposed as a vehicle for overcoming the growing problem of internet ossification [1]. This paper studies the problem of mapping diverse virtual networks onto a common physical substrate. In particular, we develop a method for mapping a virtual network onto a substrate network in a cost-efficient way, while allocating sufficient capacity to virtual network links to ensure that the virtual network can handle any traffic pattern allowed by a general set of traffic constraints. Our approach attempts to find the best topology in a family of backbone-star topologies, in which a subset of nodes constitute the backbone, and the remaining nodes each connect to the nearest backbone node. We investigate the relative cost-effectiveness of different backbone topologies on different substrate networks, under a wide range of traffic conditions. Specifically, we study how the most cost-effective topology changes as the tightness of pairwise traffic constraints and the constraints on traffic locality are varied. In general, we find that as pairwise traffic constraints are relaxed, the least-cost backbone topology becomes increasingly "tree-like". We also find that the cost of the constructed virtual networks is usually no more than 1.5 times a computed lower bound on the network cost and that the quality of solutions improves as the traffic locality gets weaker.

2006-35

Efficient Mapping of Virtual Networks onto a Shared Substrate

Authors: Jing Lu, Jonathan Turner

Corresponding Author: jl1@arl.wustl.edu

Web Page: http://www.arl.wustl.edu/~jl1

Abstract: Virtualization has been proposed as a vehicle for overcoming the growing problem of internet ossification [1]. This
paper studies the problem of mapping diverse virtual networks onto a common physical substrate. In particular, we develop a method for mapping a virtual network onto a substrate network in a cost-efficient way, while allocating sufficient capacity to virtual network links to ensure that the virtual network can handle any traffic pattern allowed by a general set of traffic constraints. Our approach attempts to find the best topology in a family of backbone-star topologies, in which a subset of nodes constitute the backbone, and the remaining nodes each connect to the nearest backbone node. We investigate the relative cost-effectiveness of different backbone topologies on different substrate networks, under a wide range of traffic conditions. Specifically, we study how the most cost-effective topology changes as the tightness of pairwise traffic constraints and the constraints on traffic locality are varied. In general, we find that as pairwise traffic constraints are relaxed, the least-cost backbone topology becomes increasingly "tree-like". We also find that the cost of the constructed virtual networks is usually no more than 1.5 times a computed lower bound on the network cost and that the quality of solutions improves as the traffic locality gets weaker.

Type of Report: Other

Efficient Mapping of Virtual Networks onto a Shared Substrate

Jing Lu and Jonathan Turner
Department of Computer Science and Engineering
Washington University in St. Louis
St. Louis, MO 63130
Email: {jl1, jst}@arl.wustl.edu

***Abstract*— Virtualization has been proposed as a vehicle for overcoming the growing problem of internet ossification [1]. This paper studies the problem of mapping diverse virtual networks onto a common physical substrate. In particular, we develop a method for mapping a virtual network onto a substrate network in a cost-efficient way, while allocating sufficient capacity to virtual network links to ensure that the virtual network can handle any traffic pattern allowed by a general set of traffic constraints. Our approach attempts to find the best topology in a family of *backbone-star* topologies, in which a subset of nodes constitute the backbone, and the remaining nodes each connect to the nearest backbone node. We investigate the relative cost-effectiveness of different backbone topologies on different substrate networks, under a wide range of traffic conditions. Specifically, we study how the most cost-effective topology changes as the tightness of pairwise traffic constraints and the constraints on traffic locality are varied. In general, we find that as pairwise traffic constraints are relaxed, the least-cost backbone topology becomes increasingly "tree-like". We also find that the cost of the constructed virtual networks is usually no more than 1.5 times a computed lower bound on the network cost and that the quality of solutions improves as the traffic locality gets weaker.**

I. Introduction

In recent years there has been a growing recognition that the protocols at the heart of the Internet have become so resistant to change that practical progress in networking has become stalled. Virtualization is widely viewed as offering a way to overcome the current impasse [1]. In a virtualized network infrastructure, diverse virtual networks share a common physical substrate consisting of both links and flexible network platforms capable of hosting multiple virtual routers [2], [3]. A *diversified* Internet can lower the barriers to entry and make it easy to deploy new network architectures and technologies, stimulating innovation and higher value services.

This paper addresses the problem of how to map virtual networks onto a common substrate in a way that enables the network to support any traffic pattern allowed by a general set of constraints, while minimizing the network cost. The problem of mapping multiple virtual networks onto a common physical infrastructure has been addressed in several different contexts. PlanetLab [4] is an overlay network testbed that is similar in spirit to the virtualized network environment that we are interested in here [2]. However, resource allocation in PlanetLab is handled in a very loose fashion. The basic operation of PlanetLab can be best characterized as "fair, best-effort." Since users can place computational demands on any node in the PlanetLab system as they choose, there is no systematic way to reserve resources for any particular purpose. In addition, PlanetLab operates purely in an overlay mode without the use of dedicated links between nodes. These factors make the nature of the resource allocation problem in PlanetLab fundamentally different from the problem we focus on here.

PlanetLab supports the use of multiple resource allocation services, which seek to balance the load across PlanetLab nodes, while satisfying objectives provided by users. One such service is *assign*, a resource discovery and allocation tool initially designed for the Emulab testbed [5]. *Assign* Characterizes resources and groups them into equivalence classes to dramatically reduce the search space. Using simulated annealing, it then seeks a good match between the user's stated resource needs and the available resources. Focusing on load balance issues on substrate networks, Zhu et al. proposed a set of heuristic algorithms for assigning substrate network resources to virtual networks [6]. Their main idea is to identify the cool spot in the substrate (i.e. an area with relatively low stress in terms of available resources) and allocate resources to the virtual networks from there.

In this paper, we focus on system contexts in which resources can be reserved for use by different virtual networks, and where resource requirements are defined in terms of a set of general traffic constraints. This is built on prior work on *constraint-based network design* [7]–[11], in which traffic is defined by a set of constraints on the traffic between designated sets of network nodes. In general, a constraint can be stated informally as "the traffic from node set S_{sub1} to node set S_{sub2} is at most B." The classical form of network design in which network traffic is specified as a matrix defining the traffic flows between each pair of nodes, can be viewed as a special case of constraint-based network design. The so-called *hose model*, popularized by Duffield [12] can also be viewed as a special case. The hose model specifies the total traffic entering and leaving each node without placing constraints on pairwise traffic flows. In this paper, we specify traffic using a combination of pairwise constraints and *termination constraints* similar to those in the hose model. We also make use of *distance-based constraints* that bound the amount of traffic between each node and its more distant peers.

Given a set of traffic constraints, the objective of constraint-based virtual network design is to find a network configuration that can handle any traffic pattern allowed by the constraints. This involves selecting a network topology comprising virtual links and virtual routers, where the links are *provisioned* with sufficient capacity to accommodate any traffic pattern permitted by the given constraints. In this paper, alternate network designs are evaluated using a cost metric in which the cost of a virtual link is proportional to the product of its capacity and its physical length. Virtual links are typically provisioned under the assumption that traffic is routed along the least-cost path in the selected virtual network topology, although other routing policies can also be handled. (It should be noted that the routes used for network dimensioning are best viewed as the "default paths" rather than the only paths that may be used in the operational network.)

Our experiments show that the system of traffic constraints has a profound influence on the least-cost network structure. In particular, tight pairwise constraints favor network topologies in which all pairs of nodes are directly connected by links with just the right capacity. While constraints get looser, "tree-like" topologies are advantageous in reducing network costs. With constraints provided by the pure hose model, the most cost-effective network topologies turn to have all nodes connected through a single, centrally located intermediate node. In addition, we find the least-cost network structure is also affected by the underlying substrate network topology.

The paper proceeds as follows: Section II describes the constraint-based virtual network design and mapping onto substrate networks and our iterative design method. In particular, II-A discusses the traffic constraints used for describing traffic flows in networks. Link dimensioning using maximum flow computations is presented in II-B, followed by a mixed integer quadratic programming formulation of the mapping problem in II-C. Section III shows how to compute a general lower bound on the configuration cost. The iterative virtual network design and mapping tool is described in Section IV. Using this tool, virtual networks with different backbone topologies under various conditions are mapped to three substrate networks. The results are discussed in Section V. Finally, we conclude our paper in Section VI.

II. Constraint-based Virtual Network Design

The virtual network design problem starts with a substrate network that is represented by an undirected graph $H = (W, F)$, in which each edge e has an associated length, and these lengths are used to define shortest path distances $d(u, v)$ between each pair of nodes u and v. For a particular virtual network we also have a set of *access nodes* $A \subseteq W$ that represents the locations in the substrate at which traffic enters or exits that virtual network. Finally, we have a specification of the expected traffic for the virtual network, which is given in the form of a set of general traffic constraints. Each of the constraints is simply an upper bound on the allowed traffic from some set $A_1 \subseteq A$ to a disjoint set $A_2 \subseteq A$.

The objective of the virtual network design problem is to select a virtual network that has sufficient capacity to handle any specific traffic pattern allowed by the given traffic constraints, while minimizing the overall use of substrate resources. A virtual network is a directed graph, defined on a subset of the substrate nodes that includes all of the access nodes, and possibly some others. Each directed edge (u, v) of the virtual network is mapped to a shortest path in the substrate and is assigned a length equal to the path length $d(u, v)$ in the substrate. Each edge (u, v) in the virtual network is also assigned a capacity $c(u, v)$, which must be sufficient to ensure that the virtual network can handle any allowed traffic pattern. To account for the use of substrate resources by a virtual network, we "charge" the virtual network an amount proportional to $c(u, v)d(u, v)$ for each virtual network link.

Virtual network link capacities can be determined using linear programming [9], for any fixed routing policy. In particular, if shortest path routing is used, one can determine for each link, the traffic pattern allowed by the given constraints that maximizes the traffic sent through the link. Thus, the key issue in virtual network design is to determine which substrate nodes to include in the virtual network and which pairs of nodes to connect with virtual links.

Since the problem of finding an optimal virtual network design is NP-hard, we seek to develop methods that produce cost-effective, if not optimal designs. One approach to this involves a direct search over the space of virtual network topologies. A given solution can be incrementally modified by adding/removing links and/or nodes, then the links of the modified topology re-dimensioned, so that the cost can be evaluated. This can be done using simulated annealing, or some similar local search technique. This approach has two drawbacks. First, to evaluate any candidate modification to the current topology, we must recompute all the link dimensions. Second, the overall huge space of candidate topologies makes it difficult to determine which of the large number of possible local modifications to choose from.

For these reasons, we explore a more structured approach that reduces the per step overhead associated with exploring alternate topologies, and effectively reduces the size of the search space that must be explored. To do this, we constrain the virtual network topologies to what we call *backbone-star* topologies. In a backbone-star topology, some of the nodes are designated as backbone nodes, while the remainder are referred to as access nodes. Each access node has a single edge connecting it to a backbone node, meaning that each backbone node is at the center of a "star" formed by its neighboring access nodes. The subset consisting of the backbone nodes can be connected together in an arbitrary fashion, but in this paper we further constrain the search space by specifying a particular backbone topology, such as a complete graph, a ring or a star. The restrictions we impose on the topology make the virtual network design problem primarily a problem of mapping the virtual network onto the substrate in the most efficient way. Fig. 1 illustrates a virtual network in a backbone-star topology, where the four backbone nodes are connected into a complete

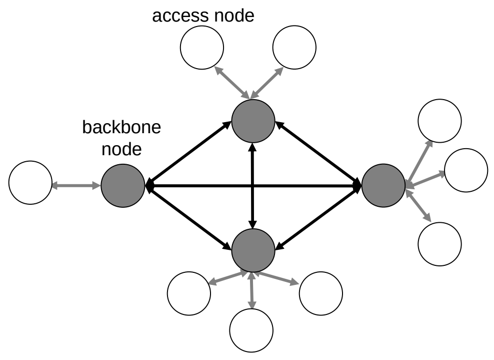

Fig. 1. Example of a virtual network in a backbone-star topology, where the four backbone nodes are connected into a complete graph

graph.

Note that if we have a fixed virtual network with dimensioned links, we can quickly evaluate alternative mappings of the backbone nodes to the substrate. However, as we change the backbone mapping, we also change the shortest paths in the virtual network, which changes the required link dimensions. These inter-dependencies have led us to adopt the iterative method outlined below.

1) *Select an initial mapping of backbone nodes onto the substrate*. This initial mapping can be arbitrary, and simply provides a starting point for the iterative refinement procedure.
2) *Connect access nodes to backbone nodes*. To provide flexibility in the virtual network topology, we do not make a rigid connection between access nodes and backbone nodes. Instead, during each iteration of the algorithm, we connect each access node to the backbone node that is closest to it in the substrate.
3) *Compute shortest paths*. Given the specified topology connecting the backbone nodes, the mapping of backbone nodes onto the substrate, and the connection of access nodes to backbone nodes, we have a complete virtual network topology with defined link lengths that we can use to compute shortest paths in the virtual network.
4) *Determine link capacities*. This can be done using linear programming as in [9] or for restricted classes of traffic constraints using maximum flow computations. This will be discussed further in II-B.
5) *Find best backbone node mapping*. The previous steps result in a complete virtual network topology with defined link capacities. We now explore alternative mappings of backbone nodes onto the substrate, while maintaining the same virtual network topology and link capacities. The best mapping found in this step is then used in the next iteration of the algorithm, which continues from step 2, above.

The computations required in steps 2 and 3 are straightforward and won't be discussed further. The computations required in steps 4 and 5 are discussed in more detail in II-B and II-C, below. The iterative procedure terminates either when there no further improvement in the quality of the solution obtained, or a pre-specified upper bound on the number of iterations is reached.

A. Defining Traffic Constraints

In general, traffic constraints can be expressed as upper bounds on the traffic between arbitrary subsets of the virtual network nodes. Although our approach can be applied to virtual networks described by arbitrary constraints, there are certain types of constraints that are particularly appropriate for describing network traffic. By imposing some structure on the system of constraints, we can make it easier for network planners to define appropriate constraints, while also reducing the computational effort required for link dimensioning.

For these reasons, we focus here on three classes of constraints that are suitable for describing traffic flows in networks. *Termination constraints* describe the total traffic terminating at the virtual network's access nodes and are described by two functions α and ω, where $\alpha(u)$ is an upper bound on the outgoing traffic from an access node u and $\omega(u)$ is an upper bound on the incoming traffic to an access node u. α and ω are also called the egress traffic and ingress traffic, respectively. When termination constraints are the only constraints specified, we have an instance of the so-called hose model [12].

Pairwise traffic constraints are specified by a function $\mu(u,v)$ which gives an upper bound on the traffic from an access node u to another access node v. We allow $\sum_v \mu(u,v)$ to exceed $\alpha(u)$ and $\sum_u \mu(u,v)$ to exceed $\omega(v)$. When $\sum_v \mu(u,v)$ is close to $\alpha(u)$ for all u and $\sum_u \mu(u,v)$ is close $\omega(v)$ for all v, we say that the pairwise constraints are tight, otherwise they are loose.

For each access node u, we define $\gamma(u)$ to be the *local neighborhood* of u. To limit the total amount of traffic at u that is permitted to leave its neighborhood, we specify the *distance constraints* by the functions α_F and ω_F, where $\alpha_F(u)$ is an upper bound on the total traffic from node u to nodes outside of $\gamma(u)$ and $\omega_F(u)$ is an upper bound on the total traffic going to node u from nodes outside of $\gamma(u)$.

Distance constraints complicate the derivation of the pairwise constraints somewhat. We now describe the precise method used to compute the pairwise constraints.

For any two nodes u and v, let

$$\begin{cases} f_1(u,v) = \dfrac{\omega(v)}{\sum\limits_{t\in\gamma(u)} \omega(t)} \cdot (\alpha(u) - \alpha_F(u)) & \text{if } v \in \gamma(u) \\ f_2(u,v) = \dfrac{\omega(v)}{\sum\limits_{t\notin\gamma(u), t\neq u} \omega(t)} \cdot \alpha_F(u) & \text{if } v \notin \gamma(u) \end{cases}$$

When $v \in \gamma(u)$, $f_1(u,v)$ represents v's fair share of u's local egress traffic among all nodes within u's neighborhood $\gamma(u)$. When $v \notin \gamma(u)$, $f_2(u,v)$ is v's fair share of u's non-local egress traffic among u's non-neighbors outside of $\gamma(u)$. f_1 and f_2 are the traffic constraints from u to v from u's perspective.

We also let

$$\begin{cases} g_1(u,v) = \frac{\alpha(u)}{\sum\limits_{t\in\gamma(v)} \alpha(t)} \cdot (\omega(v) - \omega_F(v)) & \text{if } u \in \gamma(v) \\ g_2(u,v) = \frac{\alpha(u)}{\sum\limits_{t\notin\gamma(v), t\neq v} \alpha(t)} \cdot \omega_F(v) & \text{if } u \notin \gamma(v) \end{cases}$$

When $u \in \gamma(v)$, $g_1(u,v)$ represents u's fair share of v's local ingress traffic among all nodes within v's neighborhood $\gamma(v)$. When $u \notin \gamma(v)$, $g_2(u,v)$ is u's fair share of v's non-local ingress traffic among v's non-neighbors outside of $\gamma(v)$. g_1 and g_2 are the traffic constraints from u to v from v's perspective.

Depending on whether or not u and v are neighbors, traffic from u to v is bounded by the following four cases:

$$\mu(u,v) =$$
$$\delta \cdot \begin{cases} max(f_1(u,v), g_1(u,v)) & \text{if } v \in \gamma(u),\ u \in \gamma(v) \\ max(f_1(u,v), g_2(u,v)) & \text{if } v \in \gamma(u),\ u \notin \gamma(v) \\ max(f_2(u,v), g_1(u,v)) & \text{if } v \notin \gamma(u),\ u \in \gamma(v) \\ max(f_2(u,v), g_2(u,v)) & \text{if } v \notin \gamma(u),\ u \notin \gamma(v) \end{cases}$$

where δ is called the *relaxation factor*. By setting $\delta = 1$ we tightly constrain the pairwise traffic. By allowing δ to grow larger than 1, more flexibility is allowed in the traffic distribution.

B. Determining Link Capacities

In this section, we describe the procedure for dimensioning each link, to ensure that it has sufficient capacity to handle any traffic pattern allowed by the traffic constraints.

Given: A virtual network, represented as a directed graph $G = (V,E)$, a link $\ell \in E$, a deterministic routing function $R(u,v)$ specifying the path used by traffic from u to v and a set of traffic constraints defined by the functions $[\alpha, \omega, \gamma, \alpha_F, \omega_F, \mu]$. We are also given $A \subseteq V$ as a collection of access nodes.

Find: a set of *traffic flows* $f(u,v)$ that maximizes

$$\sum_{u,v\in A, \ell\in R(u,v)} f(u,v)$$

subject to the following inequalities:

$$\begin{aligned} f(u,v) \le \mu(u,v) \quad & \forall u,v \in A \\ \sum_{v\in A} f(u,v) \le \alpha(u) \quad & \forall u \in A \\ \sum_{v\in A, v\notin\gamma(u)} f(u,v) \le \alpha_F(u) \quad & \forall u \in A \\ \sum_{u\in A} f(u,v) \le \omega(v) \quad & \forall v \in A \\ \sum_{u\in A, u\notin\gamma(v)} f(u,v) \le \omega_F(v) \quad & \forall v \in A \end{aligned}$$

The value of the objective function is the capacity needed at link ℓ to ensure that ℓ has enough capacity to handle any traffic pattern allowed by the constraints. While we could solve this problem using linear programming, it can also be formulated as a maximum flow problem, allowing for a much faster solution. Since link dimensions must be computed repeatedly for every link during each iteration of the design procedure, the use of maximum flow computations can significantly reduce the overall running time.

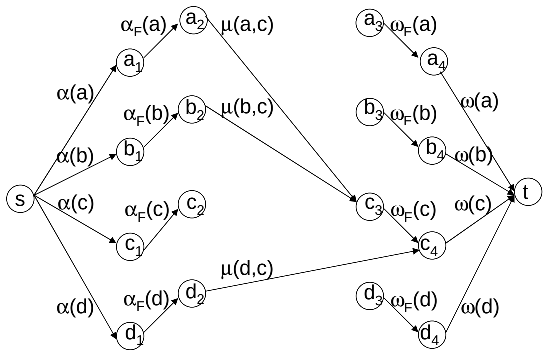

Fig. 2. Example maximum flow problem for dimensioning link ℓ

The equivalent maximum flow problem is defined on a flow graph $N = (U,M)$ where $U = \{s,t\} \cup \{u_i | u \in A, 1 \le i \le 4\}$. The edge set M includes edges of the form (s,u_1) with capacity $\alpha(u)$, edges (u_1,u_2) with capacity $\alpha_F(u)$, edges (u_3,u_4) with capacity $\omega_F(u)$, and edges (u_4,t) with capacity $\omega(u)$ for all $u \in A$. For all pairs of vertices (u,v) with $\ell \in R(u,v)$, we include an edge of the form (u_i,v_j) for $1 \le i \le 2$ and $3 \le j \le 4$ with capacity $\mu(u,v)$. Specifically, if $u \in \gamma(v)$ and $v \in \gamma(u)$ an edge (u_1,v_4) is included. If $u \notin \gamma(v)$ and $v \notin \gamma(u)$ an edge (u_2,v_3) is included. If $u \in \gamma(v)$ and $v \notin \gamma(u)$ we include an edge (u_1,v_3). If $u \notin \gamma(v)$ and $v \in \gamma(u)$ we include an edge (u_2,v_4). A maximum flow from s to t corresponds to a worst-case traffic configuration for link ℓ. The capacity limits on the edges of the form (s,u_1) and (u_4,t) ensure that the termination constraints are satisfied. The placement of the edges of the form (u_2,v_j) and (v_i,u_3) together with the capacity limits on the edges of the from (u_1,u_2) and (u_3,u_4) ensure that the distance constraints are satisfied. Finally, the capacities of the edges of the form (u_i,v_j) ensure that the pairwise constraints are satisfied. An example of one such flow graph for a virtual network with access node set $A = \{a,b,c,d\}$ is shown in Figure 2. Three pairs of nodes, (a,c), (b,c) and (d,c) have their traffic go through link ℓ.

C. Backbone Node Mapping

In this section, we describe how we map backbone nodes onto the substrate. We formulate the backbone mapping problem as a mixed integer quadratic program. We are given a substrate network $H = (W,F)$ and a virtual network $G = (V,E)$. For each vertex $u \in V$, we are also given a set of substrate vertices $t(u) \subseteq W$ that defines the set of locations that u may be mapped to. For access nodes in the virtual network, $t(u)$ will be a single substrate node, while for backbone nodes, $t(u)$ will be a subset of W. For each pair of substrate nodes (p, q), we are also given a *distance* $d(p, q)$ representing the shortest path length between nodes p and q in the substrate. Finally, for link (u, v) in the virtual network, we are given a *capacity* $c(u, v)$. In particular, the unit link capacity cost is set equal to the physical length of the link, reflecting the higher costs associated with longer links.

Given all of the above, we want to construct a least-cost mapping of virtual nodes onto substrate nodes. We represent the mapping using a collection of indicator variables $x_{u,p}$. $x_{u,p} = 1$ indicates that $u \in V$ is mapped to $p \in W$. Otherwise, $x_{u,p} = 0$. With this definition, we can define the objective function for our mixed integer quadratic program as

$$\sum_{(u,v)\in E} \sum_{p,q\in W} x_{u,p} x_{v,q} c(u, v) d(p, q)$$

To minimize the quadratic objective function, we need to specify certain constraints on the indicator variables. Specifically,

$$\begin{aligned} x_{u,p} \in \{0, 1\} &\quad \forall u \in V, p \in W \\ \sum_{p\in W} x_{u,p} = 1 &\quad \forall u \in V \\ x_{u,p} = 0 &\quad \forall u \in V, p \in W, p \notin t(u) \end{aligned}$$

We solve the problem using a general solver called MIQPBB, developed by Fletcher and Leyffer [13]. MIQPBB uses depth-first tree search and maximal fractional branching.

III. Lower Bound on Virtual Network Configuration Cost

We evaluate virtual network designs by comparing their costs to a general lower bound that is independent of the virtual network topology.

The input to the lower bound computations includes the substrate network, specified by an undirected graph $H = (W, F)$, with edge lengths and resulting shortest path distances $d(u, v)$. The input also includes a set A of access nodes, and a set of traffic constraints defined by the functions $[\alpha, \omega, \gamma, \alpha_F, \omega_F, \mu]$. Given these inputs, we seek a set of traffic flows $f(u, v)$ that maximizes

$$\sum_{u,v\in A} d(u, v) f(u, v)$$

subject to the following inequalities:

$$\begin{aligned} f(u, v) \le \mu(u, v) &\quad \forall u, v \in A \\ \sum_{v\in A} f(u, v) \le \alpha(u) &\quad \forall u \in A \\ \sum_{v\in A, v\notin\gamma(u)} f(u, v) \le \alpha_F(u) &\quad \forall u \in A \\ \sum_{u\in A} f(u, v) \le \omega(v) &\quad \forall u \in A \\ \sum_{u\in A, u\notin\gamma(v)} f(u, v) \le \omega_F(v) &\quad \forall u \in A \end{aligned}$$

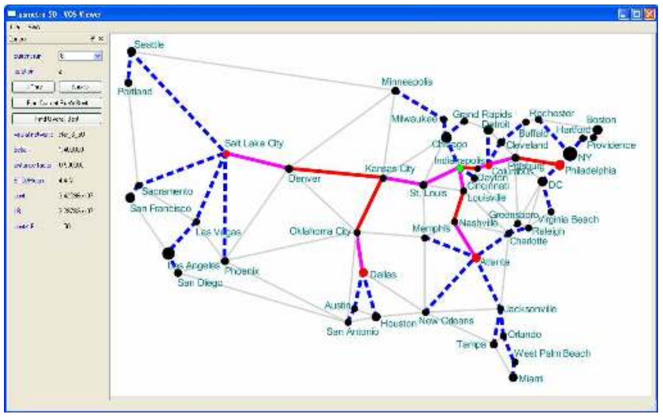

Fig. 3. GUI for viewing configuration results

This problem can be solved using linear programming. Alternatively, it can be formulated as a maximum cost flow problem, defined on a flow graph similar to the one used for the link dimensioning problem. As before, we have a network $N = (U, M)$ where $U = \{s, t\} \cup \{u_i | u \in A, 1 \le i \le 4\}$. The edge set includes edges of the form (s, u_1) with capacity $\alpha(u)$, edges (u_1, u_2) with capacity $\alpha_F(u)$, edges (u_3, u_4) with capacity $\omega_F(u)$, and edges (u_4, t) with capacity $\omega(u)$ for all $u \in A$. These edges all have zero cost. For all pairs of vertices (u, v), we include an edge of the form (u_i, v_j) for $1 \le i \le 2$ and $3 \le j \le 4$ with capacity $\mu(u, v)$ and cost $d(u, v)$. If $u \in \gamma(v)$ and $v \in \gamma(u)$ an edge (u_1, v_4) is included. If $u \notin \gamma(v)$ and $v \notin \gamma(u)$ an edge (u_2, v_3) is included. If $u \in \gamma(v)$ and $v \notin \gamma(u)$ we include an edge (u_1, v_3). If $u \notin \gamma(v)$ and $v \in \gamma(u)$ we include an edge (u_2, v_4). A maximum cost flow from s to t corresponds to a worst-case traffic configuration for any virtual network defined on this substrate with the given constraints.

IV. A Tool for Virtual Network Configurations in Diversified Networks

A tool has been implemented to automate the iterative virtual network design and mapping process. To use the tool, one specifies a virtual network backbone topology, a substrate network, a set of access nodes, and a set of traffic parameters. After an initial mapping of backbone nodes to substrate nodes is selected, the tool carries out the iterative design procedure discussed in Section II.

The tool also includes a graphical user interface that visualizes the computed virtual network configurations. For example, Fig. 3 shows the least-cost configuration of a virtual network on a substrate network consisting of the 50 largest metropolitan areas in the United States. The area of the circles shown for each node are proportional to the total population of the city represented by that node. This virtual network has 6 backbone nodes connected in a star topology with the central backbone node (shown in green) located in Indianapolis. The other 5 backbone nodes (shown in red) are in Salt Lake City, Dallas, Atlanta, Columbus, and Philadelphia. Backbone links are highlighted in red, and access links shown as dashed blue

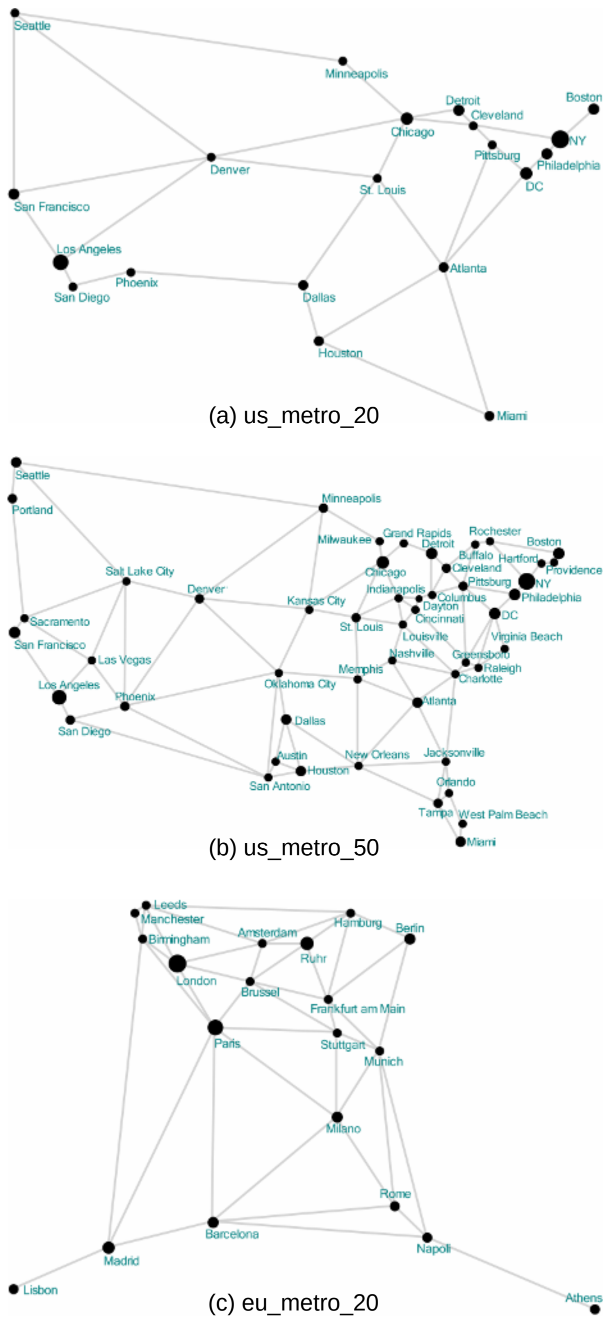

Fig. 4. Substrate networks

lines connect metro areas to their nearby routers. If a link is used both as a backbone link and an access link, it is shown in pink. On the left hand side of the window, some useful statistics associated are displayed, including the values of the traffic parameters, the cost of the configuration, the lower bound and the ratio of the configuration cost to the lower bound.

V. Evaluation

A. Experiment Setup

In this section, we describe a set of experiments carried out using the virtual network design tool described above. We consider three substrate network topologies taken from [11].

- *Us_metro_20*: this substrate network spans the 20 largest metropolitan areas in the United States.
- *Us_metro_50*: this substrate is a larger version of us_metro_20, which spans the 50 largest metropolitan areas in the United States.
- *Eu_metro_20*: this substrate spans the 20 largest metropolitan areas in western Europe.

The substrate network topologies are shown in Fig. 4. We assume substrate links have sufficient capacities to handle the traffic for the virtual network being mapped. We also assume that all substrate nodes are access nodes for each virtual network. We use the population of each access node to define the total traffic terminating at that node. That is, we define the values of the functions α and ω to be proportional to the populations of the associated metropolitan areas. For simplicity, we let $\alpha(u) = \omega(u)$ for all access nodes u. To define the distance constraints, we let the neighborhood of each node be the set of three nodes that are closest to it in the substrate. We then limit the total traffic leaving the neighborhood to be a fixed percentage of the total traffic at a node. That is, we let $\alpha_F(u) \leq \theta \cdot \alpha(u)$ and $\omega_F(u) \leq \theta \cdot \omega(u)$, for some constant θ. In our experiments, we let θ take on values 0.25, 0.5, 0.5 and 1.0. We refer to the constant θ as the *distance factor*. Note when $\theta = 1$, all traffic at a node may be non-local, in which case, we set the local neighborhood of each access node to be empty. For the relaxation factor δ, we allow it to vary from 1.0 to 1.6.

We have studied five different virtual network backbone topologies in our experiments: star, ring, star-ring, complete graph, and minimum spanning tree (MST). The star-ring topology combines a star topology with a ring, connecting the leaves of the star. For the MST topology, the set of links included in the backbone is recomputed at the start of each iteration. For each backbone topology, we also vary the number of backbone nodes. In particular, for us_metro_20 and eu_metro_20, the number of backbone nodes ranges from 3 to 10, and for us_metro_50, the number ranges from 3 to 16. In order to see the impact of the number of backbone nodes on the configuration cost, we add constraints in the mapping formulation to allow only one virtual network backbone node to be mapped to a substrate node.

The different parameter combinations generate a total of 4200 virtual network configuration problems. To help understand the analysis, we define the following terms:

- *Run*: it is a complete virtual network configuration process, which starts with a randomly selected backbone router placement, and iterates until there is no further change in placement in two consecutive iterations or a pre-set 10 iteration limit is reached. In our experiments,

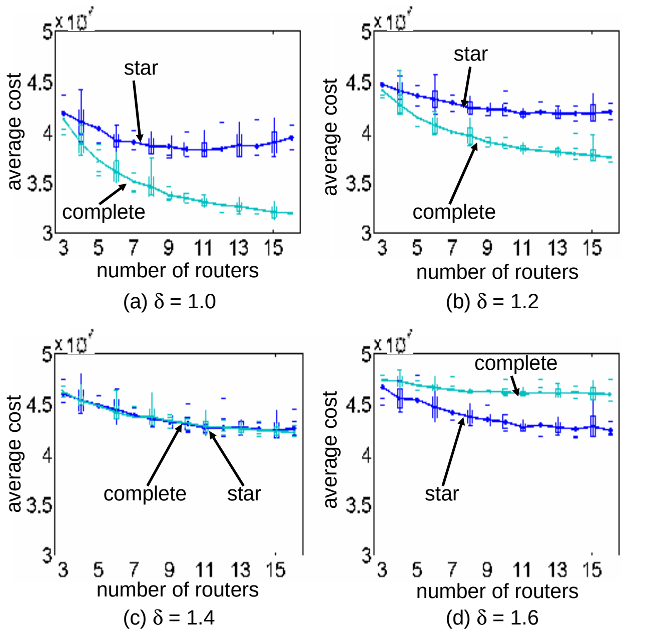

Fig. 5. Average configuration cost and error of star and complete topologies on us_metro_50 when distance factor $\theta = 0.75$

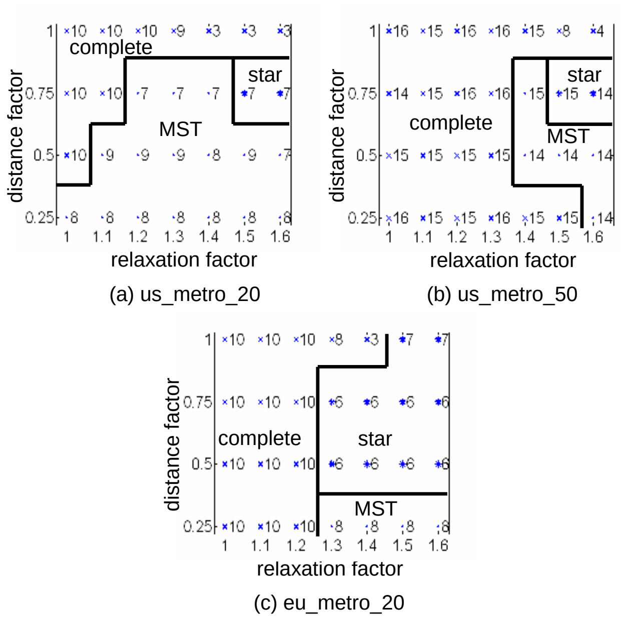

Fig. 6. Least-cost configurations on us_metro_20, us_metro_50 and eu_metro_20 under different conditions

for each virtual network configuration problem, 20 independent runs are performed, each with a different random starting point.

- *Configuration cost of a run*: a run may take multiple iterations, which are valid configurations each associated with a configuration cost. The configuration cost of a run is defined to be the cost of the least-cost configuration in that run.
- *Average configuration cost*: this is the mean value of the 20 configuration costs in 20 runs on the same virtual network.

B. Evaluation Results

Fig. 5 has four subgraphs that show the average configuration cost for the star and complete backbone topologies mapped onto us_metro_50 with a distance factor equal to 0.75. Note that each subgraph has a different relaxation factor. The x axis is the number of routers in each topology. The average costs shown are averaged over 20 runs. To evaluate the variation among these runs, we also show, for each data point, the minimum and maximum cost among the 20 runs. In addition, we compute the standard deviation and show it as a rectangle around the average cost. The standard deviation is typically within 4% to 8% of the average cost, which indicates that the randomly-chosen backbone node placement has fairly small impact on the configuration results found by the tool.

In most cases as shown in the four subgraphs, the average configuration cost decreases as the number of backbone nodes increases. However, the benefit of having more backbone nodes becomes negligible when the number of backbone nodes is sufficiently large. In some cases, increasing the number of backbone nodes even causes the configuration cost to go up.

As the relaxation factor increases, so are the configuration costs of both topologies. However, the cost of the star topology grows much slower than the cost of the complete topology. In subgraph 5a where relaxation factor $\delta = 1.0$, the complete topology is clearly better than the star. As δ increases, pairwise traffic constraints are further relaxed and the difference between the complete topology and the star gets smaller. In subgraph 5d where δ is 1.6, reflecting very loose pairwise traffic constraints, the star outperforms the complete topology.

In Fig. 6, we show the least-cost backbone topologies found under different conditions for all three substrate networks. The x and y axes are the relaxation factor and distance factor, respectively. The best backbone topology for each combination of relaxation factor and distance factor is identified by the character that precedes the number at each point in the chart. Points for which the star backbone topology is best are marked by "$*$", points for which the complete backbone topology is best are marked by "$\times$" and points for which the MST topology is best are marked by "$\cdot$". The number at each point represents the number of backbone nodes in the most cost-effective configuration. Note that the ring and star-ring backbone topologies were never the most cost-effective and so do not show up in these charts. The charts reveal both some interesting similarities and some significant differences. In general, we see that the complete graph is preferred when the pairwise constraints are tightest (small relaxation factor) while the star and MST do best when the relaxation factor is large. The MST outperforms the star when the locality constraints are stronger. In spite of these general observations, the regions of best performance for the different backbone topologies varies fairly widely. We believe this is due to the differences inherent in the underlying substrate network topologies. Here, we see

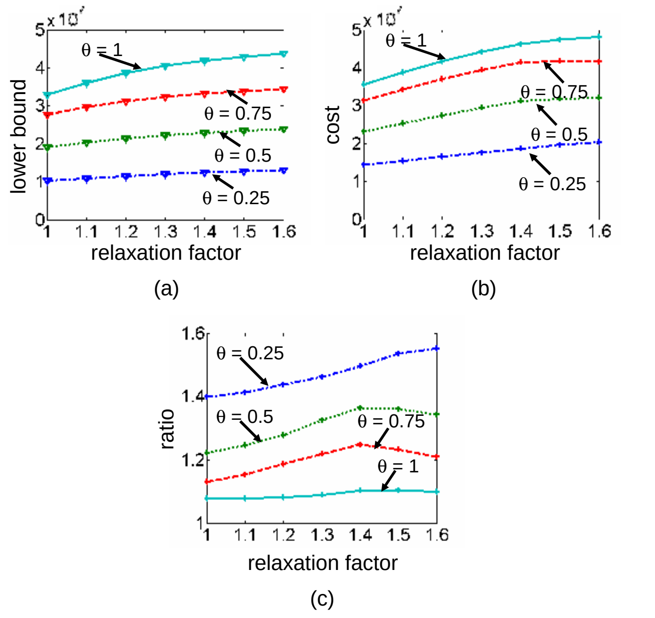

Fig. 7. Lower bound, cost of the least-cost configurations in Fig. 6, and the ratio of cost to lower bound on us_metro_50 under different conditions

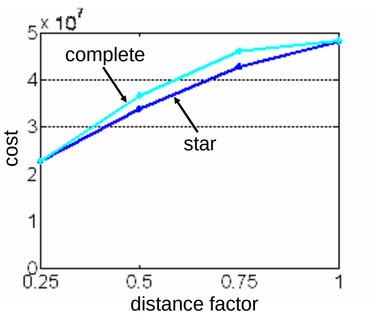

Fig. 8. Cost of the least-cost configurations for star and complete topology with 8 routers on us_metro_50 when relaxation factor = 1.6

our iterative design tool is very sensitive to the changes in traffic conditions and substrate network topologies.

In the conventional network design context, it has been shown, when traffic between pairs of end points is tightly constrained (small relaxation factor), the complete topology is optimal, and the best star is close to the optimal topology while there is only egress and ingress constraints (infinite large relaxation factor) [11]. Even though our study focuses on the backbone topology of a virtual network and is restricted by the underlying substrate topology, our results, in these cases, still conform to the observations in the conventional network design.

Fig. 7a shows how the lower bound varies as a function of the relaxation factor and distance factor for us_metro_50. We observe that cost grows as either factor increases, since the looser constraints on the traffic allow higher cost traffic configurations. Fig. 7b shows how the cost of the best virtual network configuration found varies as a function of the relaxation factor and distance factor for us_metro_50. We see that the costs vary in a similar fashion to the lower bound. Fig. 7c shows the ratio of the cost of the best configuration to the lower bound. Here we find, that the most cost-effective configurations come close to the lower bound when the locality constraints are loosest. Overall, the cost of the best virtual network configuration is no more than 1.5 times the lower bound. Also note in Fig. 7c, the two curves for $\theta = 0.5$ and $\theta = 0.75$ have peak points at relaxation factor = 1.4, which corresponds to the topology transitions indicated in Fig. 6b. The lower bound and the cost of the best virtual network configuration as a function of the two factors are not shown for us_metro_20 and eu_metro_20 because they have very similar characteristics as what we see in Fig. 7.

Fig. 8 compares the costs of the least-cost configurations for virtual networks with 8 routers in the complete and star backbone topologies. The relaxation factor is 1.6 and the substrate network is us_metro_50. When the distance factor is 0.25 or 1.0, the costs of the star and complete topology are about the same. When the distance factor is 0.5 or 0.75, the star is less expensive than the complete topology. To explain this phenomenon, we show the best configurations of the corresponding complete and star topologies in Fig. 9 and Fig. 10. We vary the distance factor while keeping the relaxation factor fixed at 1.6. When distance factor θ is 0.25, 75% of the total traffic is confined within each node's local neighborhood, which means heavy traffic on access links and light traffic on backbone links. To lower the cost on the access links, routers in both topologies are spread out in the substrate network to make access links short. When nodes within the same local neighborhood have to access the virtual network through different routers, the traffic has to pass through some backbone links. We call such traffic the *detoured local traffic*. To handle the *detoured local traffic*, direct backbone links between every pair of routers in the complete topology helps to lower the cost. As to the star topology, such traffic has to take a much longer route through the star center (highlighted in green), which contributes to a higher cost on backbone links in star topology. Even though star is more suitable than complete topology in handling the rest 25% non-local traffic, the *detoured local traffic* kills this advantage. When distance factor gets larger, increased non-local traffic starts to play a more important role in the configuration cost. To handle the increased non-local traffic, the capacities of backbone links must increase accordingly. In Fig. 9 and Fig. 10, we see the routers in both topologies retreat towards the center of the substrate network in order to lower the cost through shortening the backbone links. This is done at the expense of longer access links and hence higher costs for access links. Because the routers move towards the network center, more nodes can share a router, which largely reduces the negative impact of the *detoured local traffic* on the star topology. Due to this reason, we see star costs less than the complete topology. When θ equals 1.0 with all traffic being non-local, the routers are clustered at the network center. At this point, there is no noticeable difference in cost between the star and complete topology. The "shrinking" phenomenon is quite interesting,

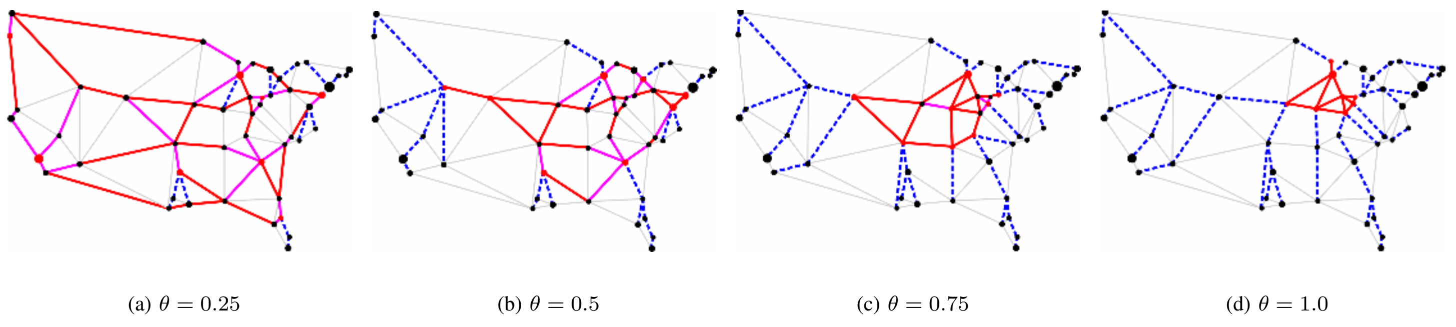
(a) $\theta = 0.25$ (b) $\theta = 0.5$ (c) $\theta = 0.75$ (d) $\theta = 1.0$

Fig. 9. Best configurations for virtual network with 8 routers in complete topology on us_metro_50 when relaxation factor $\delta = 1.6$

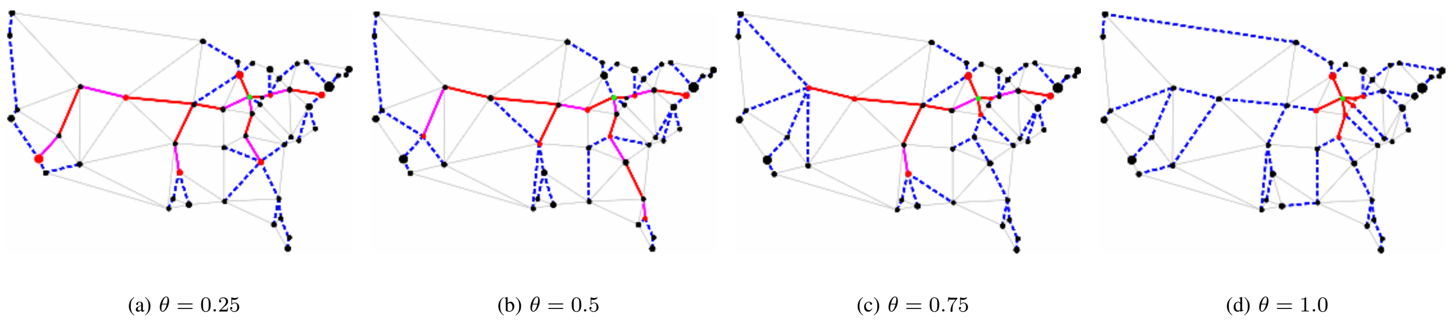
(a) $\theta = 0.25$ (b) $\theta = 0.5$ (c) $\theta = 0.75$ (d) $\theta = 1.0$

Fig. 10. Best configurations for virtual network with 8 routers in star topology on us_metro_50 when relaxation factor $\delta = 1.6$

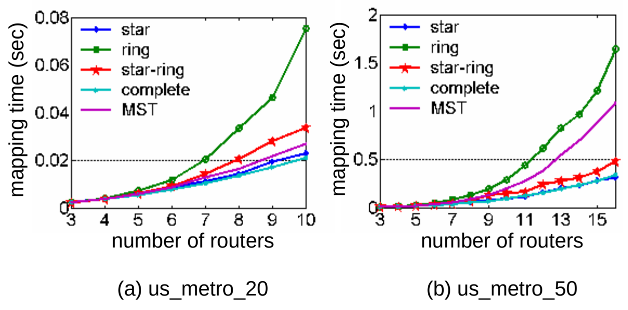

Fig. 11. Average mapping time on us_metro_20 and us_metro_50

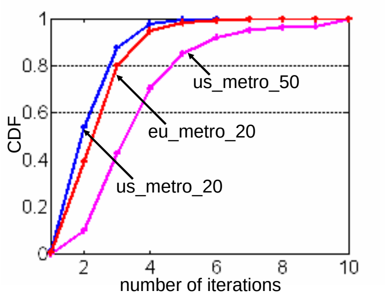

Fig. 12. CDF for the number of iterations per run on us_metro_20, us_metro_50 and eu_metro_20

which suggests, if we reduce the number of routers, we should get lower cost. This is actually proven to be true in Fig. 6b, where the complete topology with 4 routers is shown to be the least-cost one.

The running time of an iteration is dominated by the mapping time of the MIQPBB solver. Fig. 11 shows the average CPU time for mapping virtual networks in different topologies on us_metro_20 and us_metro_50. Mapping time is a function of the substrate network topology, the virtual backbone topology and the number of routers in the virtual networks. Specifically in our experiments, mapping a virtual network with 16 routers on us_metro_50 takes 1.7 seconds for the ring topology and less than 0.4 second for the star and complete topology. On us_metro_20, mapping takes less than 0.1 second. The mapping time on eu_metro_20 is very similar to the mapping time on us_metro_20. Although the mapping time increases exponentially as the number of virtual routers increases, mapping a good-sized virtual network on a large substrate network can still be done in reasonable time.

In all experiments, a run is cut off if it doesn't converge within 10 iterations. Fig. 12 shows the Cumulative Distribution Function (CDF) for the actual number of iterations performed in each run on the three substrate networks. On us_metro_20 and eu_metro_20, nearly all runs end within 6 iterations. On us_metro_50, more than 96% of the runs finish within 9 iterations. Overall, we can see that the tool is very efficient in finding a good configuration in just a few iterations.

VI. Closing Remarks

In this paper, we have developed an effective method for computing high quality mappings of virtual networks onto substrate networks. The computed virtual networks are constructed to have sufficient capacity to accommodate any traffic pattern allowed by user-specified traffic constraints. Our computational method produces high quality results that are demonstrably close to a lower bound and is fast enough to handle networks of practical size.

There are several ways in which this work can be extended. One important direction is to incorporate additional elements into the network design procedure. In particular, we currently assume that substrate links have sufficient capacity not to constrain the mapping of virtual networks. Because substrate networks are typically designed to have enough resources for accommodating multiple virtual networks, this assumption is legitimate when the number of virtual networks on the substrate isn't very large. However, adding substrate link capacity constraints is a natural and useful extension. We also do not currently account for costs associated with mapping backbone nodes to different locations. Since the location of a backbone node affects the amount of traffic passing through it, some locations will naturally require more processing resources to be allocated to a backbone node, contributing to a higher cost.

There are also other algorithmic possibilities that we have not explored. In particular, there are other alternatives that can be used for the backbone mapping procedure that may be worth exploring. Given that we need to use multiple iterations in our search for the best overall solution, it is not clear that we need to devote so much effort to finding the best mapping in each iteration. A simple local improvement algorithm (perhaps based on simulated annealing) might produce equally good results with less overall computational effort. This would allow us to handle both larger substrates and larger virtual networks.

Load balancing on substrate networks and partial reconfiguration of virtual networks are also future research directions we plan to pursue.

Acknowledgment

We would like to thank Dr. Fletcher and Dr. Leyffer for their generous help with the MIQPBB solver [13].

References

[1] L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," in *ACM Workshop on Hot Topics in Networks (HotNets)*, 2004.

[2] J. Turner and D. Taylor, "Diversifying the Internet," 2004.

[3] D. Taylor and J. Turner, "Towards a diversified Internet," Nov. 2004.

[4] Chun, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM Computer Communications Review*, vol. 33, no. 3, 2003.

[5] R. Ricci, C. Alfeld, and J. Lepreau, "A Solver for the Network Testbed Mapping Problem," *SIGCOMM Computer Communications Review*, vol. 33, no. 2, pp. 65–81, 2003.

[6] Y. Zhu and M. Ammar, "Algorithms for Assigning Substrate Network Resources to Virtual Network Components," in *IEEE Infocom*, 2006.

[7] A. Fingerhut, S. Suri, and J. Turner, "Designing Least-Cost Nonblocking Broadband Networks," *Journal of Algorithms*, pp. 287–309, 1997.

[8] ——, "Designing Minimum Cost Nonblocking Communication Networks," in *5th International Conference on Telecommunication Systems Modelling and Analysis*, Mar. 1997.

[9] A. Fingerhut, "Approximation Algorithms for Configuring Nonblocking Communication Networks," *Doctoral Dissertation, Washington University in St. Louis*, May 1994.

[10] H. Ma, I. Singh, and J. Turner, "Constraint Based Design of ATM Networks, an Experimental Study," *Technical Report, Washington University*, Apr. 1997.

[11] S. Y. Choi, "Resource Configuration and Network Design in Extensible Networks," *Doctorial Dessertation, Washington University in St. Louis*, Dec. 2003.

[12] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive, "A flexible model for resource management in virtual private networks," in *ACM SIGCOMM*, 1998, pp. 95–108.

[13] R. Fletcher and S. Leyffer, "A Mixed Integer Quadratic Programming Package," 1998.