Report Number: WUCSE-2007-44

2007

Configuring Low Cost Metanetworks on A Shared Substrate

Jing Lu and Jonathan Turner

In a diversified internet, meta-networks (“metanets?for short) share a common substrate and offer value-added services to millions of users around the globe. Therefore, configuring low-cost metanets with links having enough bandwidth to accommodate all anticipated user traffic is critical to the success of the metanets. In this paper, we propose a novel pruning algorithm that configures metanets on any given substrate in a cost-efficient way. In contrast to other testbed configuration systems, we solve the metanet configuration problem from a higher level specification and produces a network that is dimensioned to handle the specified traffic. To the best of... **Read complete abstract on page 2.**

Configuring Low Cost Metanetworks on A Shared Substrate

Jing Lu and Jonathan Turner

Complete Abstract:

In a diversified internet, meta-networks (“metanets?for short) share a common substrate and offer value-added services to millions of users around the globe. Therefore, configuring low-cost metanets with links having enough bandwidth to accommodate all anticipated user traffic is critical to the success of the metanets. In this paper, we propose a novel pruning algorithm that configures metanets on any given substrate in a cost-efficient way. In contrast to other testbed configuration systems, we solve the metanet configuration problem from a higher level specification and produces a network that is dimensioned to handle the specified traffic. To the best of our knowledge, our work is also the first one that tries to automatically determine the best metanet topology while considering network switching costs and propagation delays. We study how the best topology changes on different substrate networks as traffic conditions vary. In general, we find that as pair-wise traffic constraints and delay bounds are relaxed, the least-cost metanet topology becomes increasingly “tree-like? We also show the impact of delay bounds on the network costs under different traffic conditions. Our algorithm produces metanet configurations that are demonstrably close to the computed lower bound and is fast enough to handle substrate networks of practical size.

2007-44

Configuring Low Cost Metanetworks on A Shared Substrate

Authors: Jing Lu, Jonathan Turner

Corresponding Author: jl1@arl.wustl.edu

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

Abstract: In a diversified internet, meta-networks (“metanets?for short) share a common substrate and offer value-added services to millions of users around the globe. Therefore, configuring low-cost metanets with links having enough bandwidth to accommodate all anticipated user traffic is critical to the success of the metanets. In this paper, we propose a novel pruning algorithm that configures metanets on any given substrate in a cost-efficient way. In contrast to other testbed configuration systems, we solve the metanet configuration problem from a higher level specification and produces a network that is dimensioned to handle the specified traffic. To the best of our knowledge, our work is also the first one that tries to automatically determine the best metanet topology while considering network switching costs and propagation delays. We study how the best topology changes on different substrate networks as traffic conditions vary. In general, we find that as pair-wise traffic constraints and delay bounds are relaxed, the least-cost metanet topology becomes increasingly “tree-like? We also show the impact of delay bounds on the network costs under different traffic conditions. Our algorithm produces metanet configurations that are demonstrably close to the computed lower bound and is fast enough to handle substrate networks of practical size.

Type of Report: Other

Configuring Low Cost Metanetworks on 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*— In a diversified internet, metanetworks ("metanets" for short) share a common substrate and offer value-added services to potentially large numbers of users around the globe. Therefore, configuring low-cost metanets with links having enough bandwidth to accommodate all anticipated user traffic is crucial to the success of the metanets. In this paper, we propose a novel pruning algorithm that configures metanets on any given substrate in a cost-efficient way. In contrast to other testbed configuration systems, we solve the metanet configuration problem from a higher level specification and produce a network that is dimensioned to handle the specified traffic. To the best of our knowledge, our work is also the first one that tries to automatically determine the best network topology while considering network switching costs and propagation delays. We study how the best topology changes on different substrates as traffic conditions vary. In general, we find that as pair-wise traffic constraints and delay bounds are relaxed, the least-cost metanet topology becomes increasingly "tree-like". We also show the impact of delay bounds on the network costs under different traffic conditions. Our algorithm produces metanet configurations that are demonstrably close to the computed lower bound and is fast enough to handle substrate networks of practical size.**

I. Introduction

Virtualization has been widely recognized as a vehicle to overcome Internet ossification [1]. Virtualized network testbeds, such as Planetlab [2], GENI [3] and VINI [4], provide an early indication of how a diversified internet might be implemented and used to deliver novel network services. Virtualized infrastructures typically consist of both high bandwidth links and flexible network platforms and serve as the physical substrate shared by multiple diverse metanets.

The substrate not only provides network resources to the metanets, but it also provides access to the end hosts that are served by the metanets. Therefore, one big challenge that faces metanet planners is to design a cost-efficient metanet within the substrate that is capable of handling all anticipated traffic generated by millions of geographically distributed end hosts. This involves choosing the right metanet topology, mapping it onto the substrate and dimensioning each metanet link ("metalink" for short) with sufficient capacity to handle the expected traffic while achieving the minimum network cost. The inter-dependencies among the metanet topology, mapping and dimensioning metalinks have made the metanet design problem intractable. One common way to tackle the problem is to try different topologies and mappings and pick the one that yields the lowest cost. In [5], we presented an iterative approach that automates the mapping and link dimensioning processes, allowing quick evaluation of different metanet topologies. Since it is infeasible to try out all possible topologies, the essential drawback of such methods is that the quality of the design highly depends on the choice of the candidate metanet topologies under evaluation.

To address this issue, we have designed a network planning tool that automatically determines the topology for a cost-efficient metanet configuration and provisions the metalinks with sufficient bandwidth to accommodate any traffic pattern allowed by a general set of traffic constraints. Our planning tool accepts general information about the substrate network, the metanet user locations and the traffic demands expressed by a set of constraints. Time-sensitive metanets may also specify propagation delay bounds between user locations. The core of the planning tool is a novel pruning algorithm that identifies the cost-efficient metanet topology and configures it on the given substrate in a systematic way.

We conduct extensive experiments on three substrates spanning the 20 and 50 largest metropolitan areas in the United States and western Europe. We vary the traffic constraint parameters and study how the least-cost metanet topology is related to the traffic constraints. Our results indicate that the system of traffic constraints has a profound influence on the least-cost network structure. In particular, tight pair-wise constraints favor network topologies in which all pairs of nodes are directly connected by links with just the right capacity. As constraints get looser, "tree-like" topologies are better at reducing network costs. Our planning tool also makes it easy to study the impact of different latency constraints on metanet topologies and costs, allowing planners to better evaluate the cost/latency tradeoffs. Overall, the costs of the metanet configurations found by our tool are no more than 1.55 times the computed lower bound.

II. Related Work

The problem of configuring a metanet on a common physical infrastructure has been addressed in several different contexts. PlanetLab [2] is a virtualized overlay network testbed. It supports multiple resource discovery and allocation services including *Assign* [6], *NetFinder* [7] and *SWORD* [8], which seek to balance the load across PlanetLab nodes, while

satisfying users' objectives. To find a good match between the user's stated resource needs and the available testbed resources, these services all require network providers to submit detailed resource needs for their overlay networks, such as the number of nodes in the overlay, the network topology, the minimum bandwidth per link, and the minimum free computation resource per node. These resource requirements may be suitable for controlled network experiments in which networks are configured for particular testing purposes with traffic generated mainly by the experimenters, but less suitable for configuring long-lived global-scale metanets that are intended to serve large number of real users. With substrate networks capable of provisioning metalinks with guaranteed bandwidth, it is more appropriate to have metanet planners provide higher level traffic specification as the driving force in metanet configuration and let the tool determine the best choice of nodes and links to satisfy the specification.

Our work bears similarities with prior work on conventional *constraint-based network design* [9]–[13], in which a network is dimensioned to satisfy constraints on the traffic between designated sets of network nodes. However, our work extends the prior work in several ways. First, it is able to automatically determine the best network topology, while the previous studies left it to the human network planners to choose the topology through an interactive process. Second, this work accounts for the switching costs of the network nodes which were neglected in the previous work. Finally, our work incorporates user-specified bounds on network latency, allowing network planners to ensure acceptable delays on all network paths.

III. Constraint-based Metanet Design

The metanet design problem starts with a substrate network represented by a single undirected graph, in which each node has resources for implementing metanet routers and is usually viewed as a traffic aggregation point for the users to access a particular metanet; each edge has an associated length equal to the physical distance spanned by the substrate link. Similarly, a metanet can be represented as a directed graph with metanodes that route traffic among user locations and metalinks that connect metenodes. A metanode can perform either local switching which routes traffic among users who access the metanet at that location or transit switching that routes traffic passing through the metanode. To distinguish the two types of metanodes based on their switching characteristics, we adopt the terminology *access node* for the metanode performing local switching and *backbone node* for the metanode performing transit switching. A metalink between an access node and a backbone node allows users to communicate with the rest of the metanet users through a designated access node. Although we find it convenient to refer to the local switching and transit switching functions as two distinct elements, in practice they will often be implemented as part of a single system. Therefore, when an access node and a backbone node are hosted at the same substrate node, the cost of the metalink between them is ignored. Fig. 1 illustrates a portion of a metanet embedded in a substrate with four metanodes on three substrate nodes. Each of the two access nodes connects to its local users and a backbone node through a metalink.

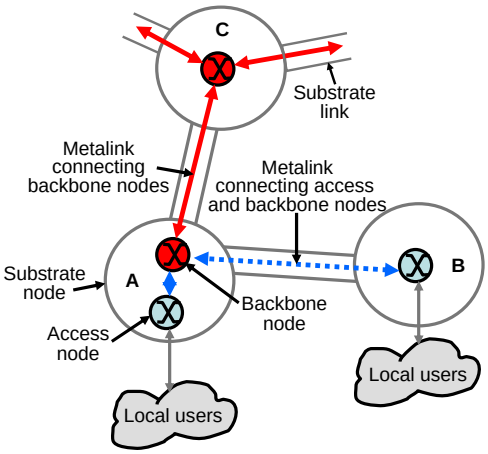


Fig. 1. A metanet embedded in a substrate. Four metanodes including two access nodes and two backbone nodes are mapped to three substrate nodes.

Part of the metanet configuration problem is to decide the metanet topology and embed it in the given substrate, with each metanode mapped to a substrate node and each metalink mapped to a simple path in the substrate with a length equal to the path length. Once an embedding is decided, each metalink can be dimensioned with a capacity that is big enough to handle any traffic pattern allowed by the specified traffic constraints. In this section, we discuss in detail the metalink dimensioning method given a set of traffic constraints and a known metanet embedding, and answer the question of how to find the least-cost embedding in Section IV.

A. Traffic Constraints

In general, traffic constraints can be expressed as upper bounds on the traffic between arbitrary subsets of the access nodes. Although our approach can be applied to metanets 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 on three classes of constraints that are suitable for describing traffic flows in networks.

1) *Termination constraints* specify the total traffic terminating at the metanet 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 u. When termination constraints are the only constraints specified, we have an instance of the so-called hose model [14].
2) *Pair-wise constraints* are specified by a function $\mu(u, v)$ which bounds the traffic from access node u to access node v. The pair-wise constraints are considered

tight 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.

3) *Distance constraints* are used to limit the amount of traffic between an access node and its more distant peers. For each access node u, $\gamma(u)$ is the *local neighborhood* of u, and $\alpha_F(u)$ 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 $\gamma(u)$. (The subscript '$_F$' stands for "far".)

Although our algorithm accepts all three classes of traffic constraints, it is up to the metanet planners to decide which type(s) of constraints best characterize the traffic in their metanets. In particular, the three classes of constraints are not completely independent, which means the metanet planners must consider them as a whole while selecting the parameters associated with each constraint. In Section VI, we provide details on how we derive the constraint parameters in our experiments.

B. Dimensioning metalinks

The pruning algorithm described in Section IV finds a candidate embedding of a metanet on a given substrate in each step and evaluates its quality by dimensioning each metalink and computing the cost of the resulting intermediate metanet configuration. Here, we describe the metalink dimensioning method, which guarantees that the provisioned bandwidth capacity is sufficient to handle any traffic pattern allowed by the traffic constraints. This is an extension to the method used in the extensible network design [12]. The problem is formally stated as follows:

Given: A metanet, 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, a set of traffic constraints defined by the functions $[\alpha, \omega, \gamma, \alpha_F, \omega_F, \mu]$, and a collection of access nodes $A \subseteq V$.

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 metalink 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. Details about how we construct the maximum flow problem can be found in [15].

C. Metanet Cost

To account for the substrate resources taken by a metanet, we define the following cost metric:

$$\begin{aligned}
C_{net} &= C_{link} + C_{node} \\
&= \rho_l \cdot \sum_{metalink\ l} length(l) \cdot bw_capacity(l) \\
&\quad + \rho_n \cdot \sum_{backbone\ node\ u} sw_capacity(u) \qquad (1)
\end{aligned}$$

The metanet cost C_{net} includes two parts: the metalink cost C_{link} and metanode cost C_{node}. For each metalink, a metanet is "charged" with an amount proportional to the product of the link length and the provisioned bandwidth capacity. Since the switching capacity of a metanode is a good indication of the amount of computational resources needed by the node, we "charge" a metanet with the switching capacity provisioned to all backbone nodes. We do not include the switching cost of the access nodes in the metric since this cost is not related to the backbone configuration and is highly dependent on the degree of traffic concentration at the access nodes. ρ_l and ρ_n are cost scaling factors for the two substrate resources.

IV. Pruning Algorithm

Traffic constraints have a direct impact on the choice of network topology. So the objective of the pruning algorithm is to find a network topology and its embedding in a given substrate to best reflect the traffic constraints. The pruning algorithm starts with a metanet that includes an access node at each user location (represented by a substrate node at the location) and a backbone node at each substrate node, and uses all the substrate links to route its traffic. It then proceeds to reduce the network cost by first pruning substrate links and then pruning backbone nodes.

A. Link Pruning

Given the traffic constraints and a substrate graph $G_s = (V_s, E_s)$, the goal of the link pruning stage is to identify a subset of substrate links $E_r \subseteq E_s$ to be used by metanet traffic and a subset of substrate nodes $V_b \subseteq V_s$ to host metanet backbone nodes. The algorithm starts with $E_r = E_s$ and $V_b = V_s$. In each pruning step, for each e in E_r, if removing e does not make the corresponding metanet disconnect or violate the predefined delay bounds, the cost of the metanet with e omitted is computed. We then "prune" the link that yields the largest cost reduction relative to the current cost, permanently removing it from E_r. The pruning step is repeated until no further improvement in network cost is possible.

The pseudo-code for the link pruning stage is shown in Algorithm 1. Some of the function calls are explained below:

1) $meta\_topology()$ constructs a metanet topology based on the access node set V_a, the current backbone node set V_b and the remaining substrate links E_r. Metalinks are

Algorithm 1: Link Prunning

Input:
Substrate topology: $G_s = (V_s, E_s)$;
Metanet access node set: $V_a \subseteq V_s$;
Metanet traffic constraint parameters: $\alpha, \omega, \gamma, \alpha_F, \omega_F, \mu$;
Metanet delay bound function: $f(u, v)$ $\forall u, v \in V_a$ and $u \neq v$;
Output:
$E_r \subseteq E_s$ and $V_b \subseteq V_s$;

$E_r = E_s$; $V_b = V_s$;
$G_m = meta\_topology(V_a, V_b, E_r)$;
$link\_dimension(G_m, \alpha, \omega, \gamma, \alpha_F, \omega_F, \mu)$;
$C_{net} = cost(G_m)$;
while TRUE do
  $C_{net_i} = \infty$; $e_i = \phi$;
  for $e \in E_r$ **do**
    $E_r = E_r - e$;
    $V_b = update\_V_b(E_r)$;
    $G_m = meta\_topology(V_a, V_b, E_r)$;
    if $violate\_delay\_bounds(f, G_m)$ = TRUE
      $E_r = E_r \cup e$;
      continue;
    end if
    $link\_dimension(G_m, \alpha, \omega, \gamma, \alpha_F, \omega_F, \mu)$;
    $C_{net_e} = cost(G_m)$;
    if $C_{net_e} < C_{net_i}$
      $e_i = e$; $C_{net_i} = C_{net_e}$;
    end if
    $E_r = E_r \cup e$;
  end for
  if $C_{net_i} \leq C_{net}$
    $E_r = E_r - e_i$; $C_{net} = C_{net_i}$;
    continue;
  else
    break;
  end if
end while
$V_b = update\_V_b(E_r)$;

Algorithm 2: Node Prunning

Input:
Substrate link set after link pruning: E_r;
Metanet backbone node set after link pruning: V_b;
Metanet access node set: $V_a \subseteq V_s$;
Metanet traffic constraint parameters: $\alpha, \omega, \gamma, \alpha_F, \omega_F, \mu$;
Metanet delay bound function: $f(u, v)$ $\forall u, v \in V_a$ and $u \neq v$;
Output:
Metanet topology : $G_m = (V_m, E_m, C)$ with $V_m = V_a \cup V_b$.
$\forall (u, v) \in E_m$, (u, v) is a simple path defined on E_r between u and v, and $C(u, v)$ is the provisioned bandwidth capacity;
Metanet cost: C_{net};

$G_m = meta\_topology(V_a, V_b, E_r)$;
$link\_dimension(G_m, \alpha, \omega, \gamma, \alpha_F, \omega_F, \mu)$;
$C_{net} = cost(G_m)$;
while TRUE do
  $C_{net_i} = \infty$; $v_i = \phi$;
  for $v \in V_b$ **do**
    $V_b = V_b - v$;
    $G_{m_v} = meta\_topology(V_a, V_b, E_r)$;
    if $violate\_delay\_bounds(f, G_{m_v})$ = TRUE
      $V_b = V_b \cup v$;
      continue;
    end if
    $link\_dimension(G_{m_v}, \alpha, \omega, \gamma, \alpha_F, \omega_F, \mu)$;
    $C_{net_v} = cost(G_m)$;
    if $C_{net_v} < C_{net_i}$
      $v_i = v$; $C_{net_i} = C_{net_v}$; $G_{m_i} = G_{m_v}$;
    end if
    $V_b = V_b \cup v$;
  end for
  if $C_{net_i} \leq C_{net}$
    $V_b = V_b - v_i$; $C_{net} = C_{net_i}$; $G_m = G_{m_i}$;
    continue;
  else
    break;
  end if
end while

established among metanodes using the shortest paths in the portion of the substrate defined by E_r. Specifically, each access node in V_a has a metalink connecting it to the closest backbone node in V_b. Two backbone nodes u and v have a metalink connecting them if and only if no other node in V_b lies on the shortest path between u and v. This method is simple and intuitive. It reduces link redundancy and lowers link cost.

2) $link\_dimension()$ dimensions each of the metalinks to handle the traffic defined by the traffic constraints.
3) $cost()$ computes the metanet cost using Formula (1).
4) $update\_V_b()$ removes nodes from V_b after links are pruned from E_r. A node v is removed either when v is no longer connected by any link in E_r, or when v is also an access node and has only one link in E_r incident to it. In the first case, v is removed because no traffic can reach v. In the second case, v is removed to achieve a lower node cost without increasing the link cost.
5) $violate\_delay\_bounds()$ checks to see if the metanet violates the predefined delay bounds. It also serves as a check that the metanet graph is connected.

B. Node Pruning

The second stage of the pruning algorithm works on the pruned substrate graph defined by E_r and successively removes backbone nodes from V_b, until no further improvement in network cost is possible. In each node pruning step, for each node v in V_b, we compute the metanet cost with v removed and select the node that yields the largest cost reduction relative to the current cost, and permanently remove it from V_b. The pseudo-code is shown in Algorithm 2. When the node pruning stops, we obtain a low-cost metanet configuration that satisfies both the traffic constraints and delay bound constraints.

We illustrate the pruning algorithm in the following example. Fig. 2 shows a substrate network that spans 10 metropolitan areas in the western United States. A metanet provider wants to construct a metanet to connect users in Seattle, San Francisco, Phoenix, St. Louis and Minneapolis, and also specifies a set of traffic constraints that must be satisfied by the metanet.

As shown in Fig 3, the link pruning stage starts with a metanet having an access node on each user location and a backbone node on each substrate node. Red and light blue colors are used to distinguish backbone nodes from access nodes. We also draw the metalinks between backbone nodes using red solid lines and those between an access node and a backbone node using blue dashed lines. Fig. 4 shows the embedded metanet when the link pruning stops. As a result, we have five backbone nodes located at five substrate nodes and

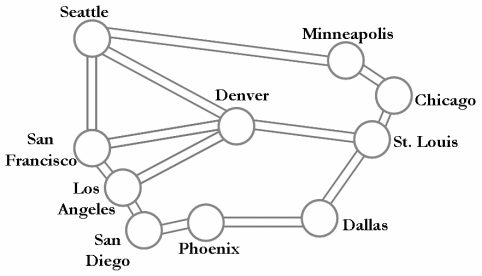


Fig. 2. A substrate network spanning 10 metropolitan areas.

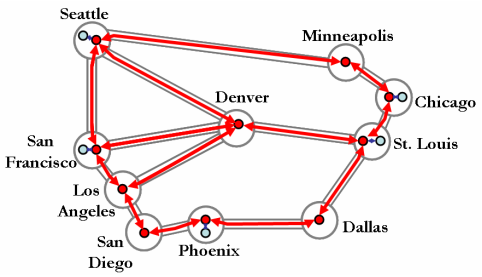


Fig. 3. The embedded metanet before link pruning. Cost = $32,000.

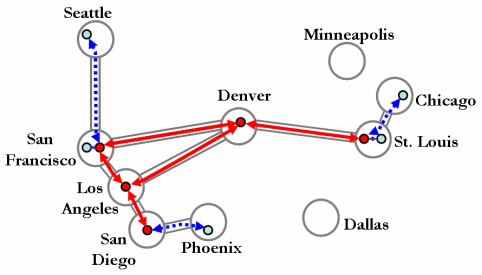


Fig. 4. The embedded metanet after link pruning. Cost = $20,000.

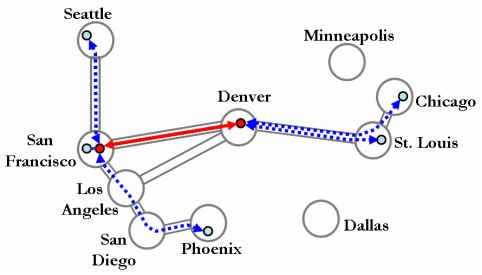


Fig. 5. The embedded metanet after node pruning. Cost = $14,000.

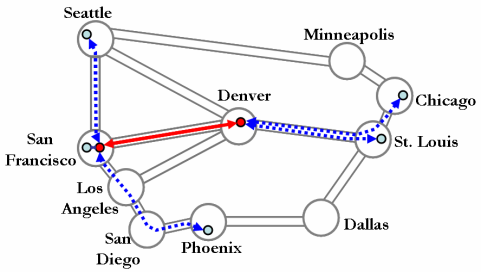


Fig. 6. The least-cost metanet configuration on the original substrate in Fig. 2. Cost = $14,000, Lower bound = $12,600.

each of the access nodes connecting to the closest backbone node through a metalink defined on the remaining substrate links. Fig. 5 shows only two backbone nodes remain when the node pruning stops. We also show the final least-cost metanet configuration on the original substrate in Fig. 6. Note that the final metanet cost is reduced significantly as the result of pruning, and is only 11% higher than the lower bound.

V. Lower Bound on Metanet Cost

In this section, we show how a general lower bound is computed on the metanet cost defined in Formula (1).

A. Lower bound on C_{link}

The idea of computing a lower bound on link cost is to find the most expensive traffic configuration on the given substrate that satisfies all the traffic constraints [9], [12]. To compute the lower bound, we are given a substrate network $H = (W, F)$ with shortest path distances $d(u, v)$ between any two nodes u and v in W, a set of access nodes $A \subseteq W$, and a set of traffic constraints defined by the functions $[\alpha, \omega, \gamma, \alpha_F, \omega_F, \mu]$. Our goal is to seek a set of traffic flows $f(u, v)$ that maximizes the link cost

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

subject to the following inequalities:

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

This linear program can be formulated as a maximum cost flow problem, defined on a flow graph similar to the one used in the link dimensioning problem. In [15], we have the detailed description on how we construct the equivalent maximum cost flow problem.

B. Lower bound on C_{node}

The lower bound on the node switching cost is derived from a simple observation. For each access node, its termination traffic is switched by at least one backbone node, so the total traffic switched by all backbone nodes is at least the total termination traffic at all access nodes. Therefore, the lower bound on C_{node} is simply $\rho_n \cdot \sum_{access\ node\ u} (\alpha(u) + \omega(u))$.

Because the lower bounds on C_{link} and C_{node} are independent of each other, we can add them together to obtain a general lower bound on the metanet cost C_{net}. Note that this general lower bound is also independent of the metanet topology and can be used to evaluate candidate metanet configurations.

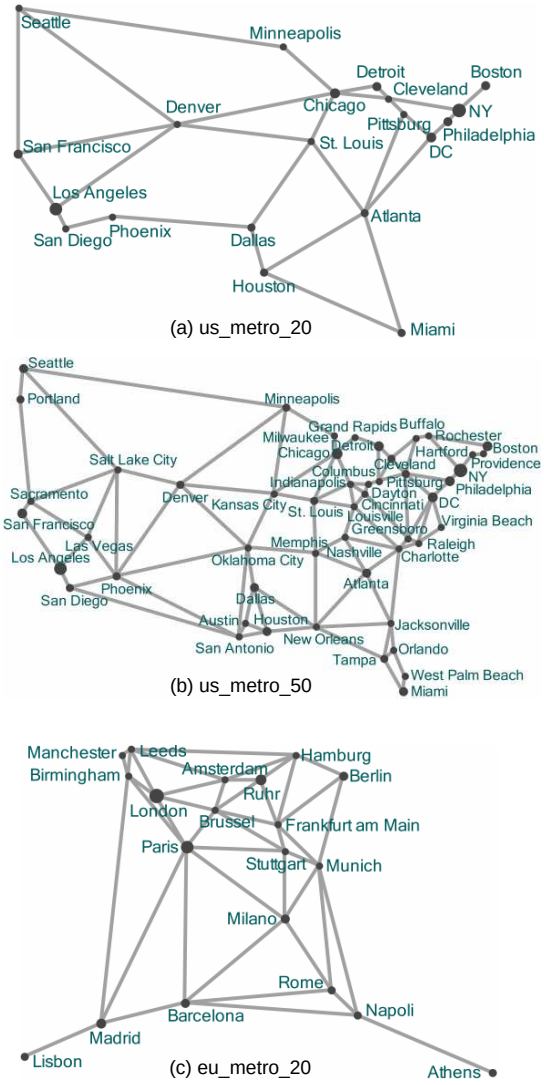


Fig. 7. Substrate networks

VI. Evaluation

A. Experiment Setup

In this section, we describe a set of experiments carried out using our metanet planning tool. We consider three substrate topologies taken from [12].

- *Us_metro_20* spans the 20 largest metropolitan areas in the United States.
- *Us_metro_50* spans the 50 largest metropolitan areas in the United States.
- *Eu_metro_20* spans the 20 largest metropolitan areas in western Europe.

The substrate network topologies are shown in Fig. 7. For a metanet to be configured on a substrate, we assume its users are at all substrate nodes. We define α and ω to be proportional to the populations of the associated metropolitan areas, and let $\alpha(u) = \omega(u)$ for each access node u. For the distance constraints, we let the local neighborhood of each access node be its three closest neighboring nodes, and limit the total traffic leaving its neighborhood to be a fixed percentage of its total termination traffic. That is, we let $\alpha_F(u) = \theta \cdot \alpha(u)$ and $\omega_F(u) = \theta \cdot \omega(u)$, for a *distance factor* $\theta \leq 1.0$. In our experiments, we let θ take on values 0.25, 0.5, 0.75 and 1.0.

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

For any two nodes u and v, let

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

When $v \in \gamma(u)$, $f_1(u,v)$ represents v's fair share of u's local outgoing 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 outgoing 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. Similarly, we derive the traffic constraints g_1 and g_2 from v's perspective, and we have

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

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 $\delta = 1.0$ we tightly constrain the pair-wise traffic. Having a δ larger than 1.0 allows more flexibility in the traffic distribution. In our experiments, δ varies from 1.0 to 1.6.

We also study the impact of the traffic propagation delay bounds on the cost of metanet configurations. For simplicity, we use the distance travelled by the traffic instead of the time to bound the propagation delay. We define the acceptable delay between u and v to be within its default minimum delay $min\_d(u,v)$ which is the shortest path distance between them in the substrate and its maximum allowed delay $max\_d(u,v)$ defined as follows:

$$max\_d(u,v) = (1 + l_2 - l_1) \cdot min\_d(u,v) + l_1 \cdot d_m \qquad (2)$$

l_1 and l_2 are scaling factors used to adjust the delay bounds, and d_m is the largest min_d among all node pairs. Fig. 8 shows the acceptable delay bounded by the functions min_d and max_d. When l_1 and l_2 are small, delay bounds are tight. Increasing l_1 allows longer delays between node pairs that are closer in the substrate, while increasing l_2 loosens the delay bounds for node pairs that are far away in the substrate.

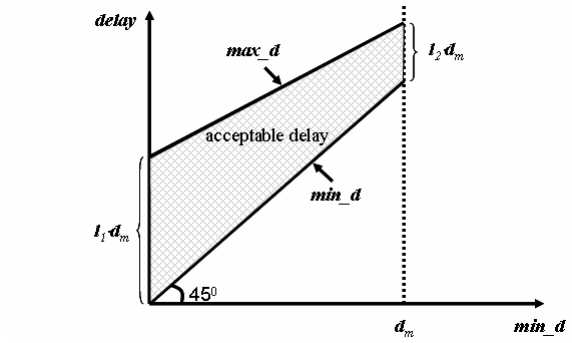


Fig. 8. Acceptable delays between the default minimum delay and the maximum delay defined in Formula (2)

Finally, for illustration purpose, we set cost scaling factors ρ_l and ρ_n to the values that are equivalent to charging $\$1$ per 100km metalink of 1Mbps provisioned bandwidth and $\$1$ per 1Mbps switching capacity.

B. Evaluation Results

1) Least-cost metanet topology: In the conventional constraint-based network design, finding the optimal topology for an arbitrary set of traffic constraints is a hard problem. However, it's well known that the complete graph is the optimal topology for networks with tight pair-wise constraints, and the best star topology is no more than twice as expensive as the optimal topology given only termination constraints [10]. Interestingly, the least-cost metanet topologies found by our pruning algorithm exhibit similar characteristics, even though the embedded metanets are restricted by the underlying substrate topologies and the more complex constraints.

Our first set of experiments focus on the impact of the tightness of the pair-wise constraints on the metanet topologies. To do so, we place no delay constraints on the access node pairs and no restriction on traffic locality. Fig. 9 shows the least-cost metanet topologies on the three substrates with tight pair-wise constraints ($\delta = 1.0$). We color a substrate node in red to indicate that a backbone node is mapped to it. We also use red solid lines to highlight metalinks connecting backbone nodes (we call them backbone links), and blue dashed lines for metalinks connecting access nodes and backbone nodes (we call them access links). If a substrate link is shared by both a backbone link and an access link, we distinguish it using a thicker red line. Note that the metanet topologies in Fig. 9 are very close to complete graphs, in which backbone nodes are well connected by mostly directed backbone links.

As we loosen the pair-wise constraints by setting the relaxation factor δ to 1.6, it's equivalent to having only the termination constraints. Fig. 10 shows the least-cost metanet topologies on the three substrates. The topologies clearly exhibit the star structure with all access nodes connected to a centrally located backbone node.

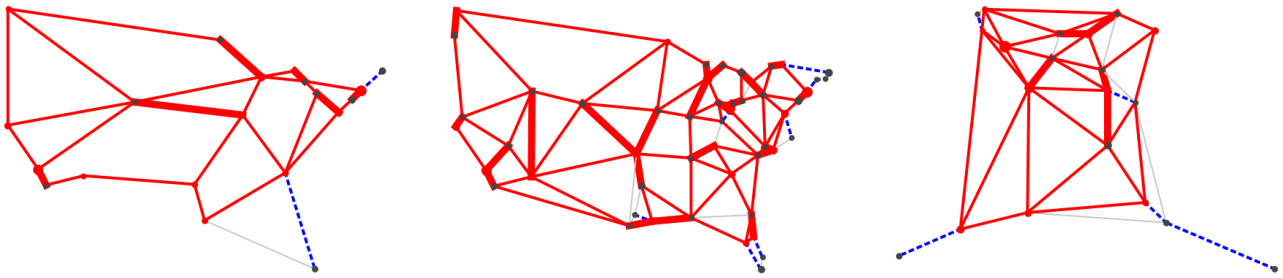
Next, we study the role that the distance constraints play in determining the least-cost metanet topologies. This time we keep loose pair-wise constraints in our experiments. Compared to Fig. 10, Fig. 11 shows dramatic topology changes as we allow no more than 25% of a node's total traffic to leave its local neighborhood. With most traffic kept local, we see the least-cost metanet topology tends to have many backbone nodes that spread across the substrate. These backbone nodes are also located close to the access nodes to provide good local connectivity.

Comparing Fig. 9 with Fig. 11, we see that the tight pair-wise constraints have a similar effect in shaping the least-cost metanet topologies as the tight distance constraints. In both circumstances, the least-cost metanet topology tends to have many backbone nodes and directly connected backbone links. However, there are also noticeable differences in the topologies. In the case of the tight pair-wise constraints, more backbone nodes are needed to provide shorter routing paths between all node pairs. In Fig. 11, there are two forces that determine the topology of the metanet. On the one hand, the loose pair-wise constraints favor aggregation of traffic using fewer backbone links and nodes just as we see in Fig. 10. On the other hand, the tight distance constraints and small local neighborhood "drag" backbone nodes close to the access nodes to provide good local connectivity. Because of these two opposing forces, some star-like clusters appear at the edges of the network and fewer backbone nodes are needed in Fig. 11 compared to Fig. 9.

2) Metanet Costs: Fig. 12 shows how the lower bound on the metanet cost varies as a function of the relaxation factor and distance factor for the three substrates. Because looser constraints allow more expensive traffic configurations, the lower bound grows as either factor increases. We also see in Fig. 13 that the metanet cost grows in a similar fashion as the lower bound. However, we notice that in Fig. 13(a) and Fig. 13(c), the growth of the metanet cost slowly levels off as the relaxation factor gets bigger. This is because when the pair-wise constraints get looser, their impact on the metanet topology as well as the metanet cost diminishes. Fig. 14 shows the ratio of the metanet cost to the corresponding lower bound. Overall, the cost of the metanet configuration is no more than 1.55 times the lower bound, and the quality of the configurations improves dramatically as the constraints get weaker.

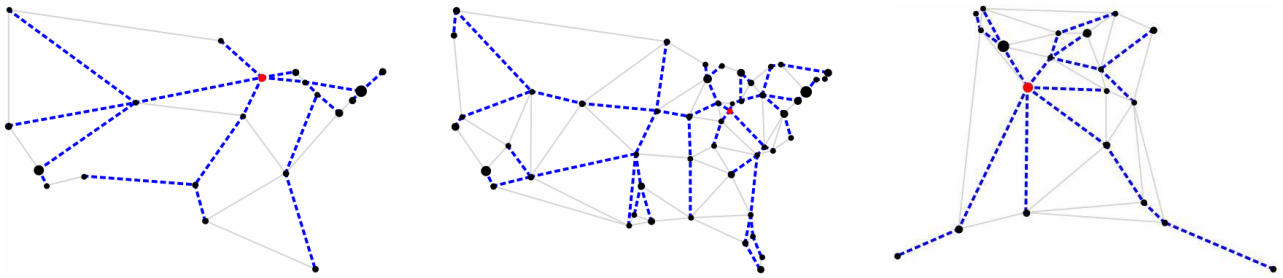
3) Impact of Delay Bounds on Metanet Costs: We see in Section VI-B.1 that loose traffic constraints favor metanet topologies with only a handful of backbone nodes located near the center of the substrate. Even though such topologies minimize the network cost and have negligible impact on the communication latencies between far-away nodes, they can dramatically increase the delay between close-by nodes, which may not be acceptable to time sensitive applications.

To study the influence of the delay bounds on the metanet costs, we first focus on the metanets with loose traffic constraints. Given the results in Section VI-B.1, we expect to see that relaxing the delay bounds on close-by nodes will allow cheaper metanet configurations. In Fig. 15, not surprisingly, as we fix the delay parameter l_2 at 0.2 and gradually increase l_1 from 0.2 to 1.4 to allow bigger delays between access



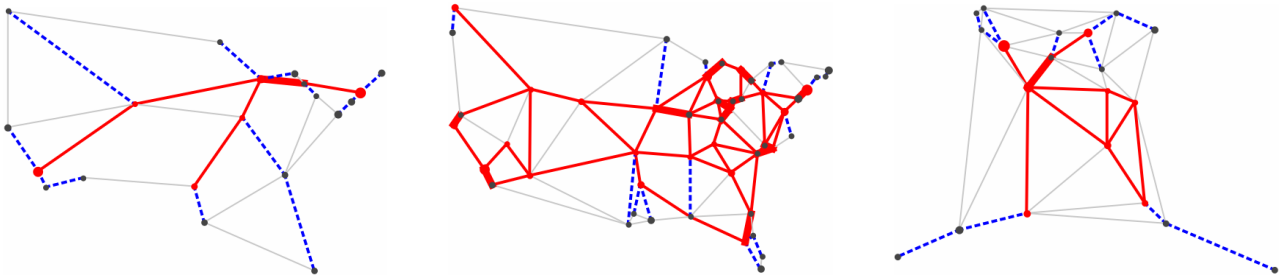
(a) US_metro_20, cost/lb = 1.12 (b) US_metro_50, cost/lb = 1.16 (c) EU_metro_20, cost/lb = 1.16

Fig. 9. The least-cost metanets on three substrates with $\delta = 1.0$ and $\theta = 1.0$



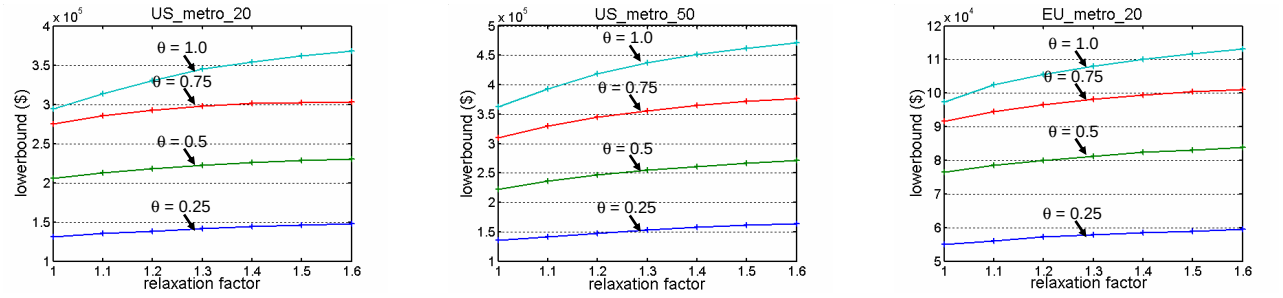
(a) US_metro_20, cost/lb = 1.07 (b) US_metro_50, cost/lb = 1.15 (c) EU_metro_20, cost/lb = 1.08

Fig. 10. The least-cost metanets on three substrates with $\delta = 1.6$ and $\theta = 1.0$



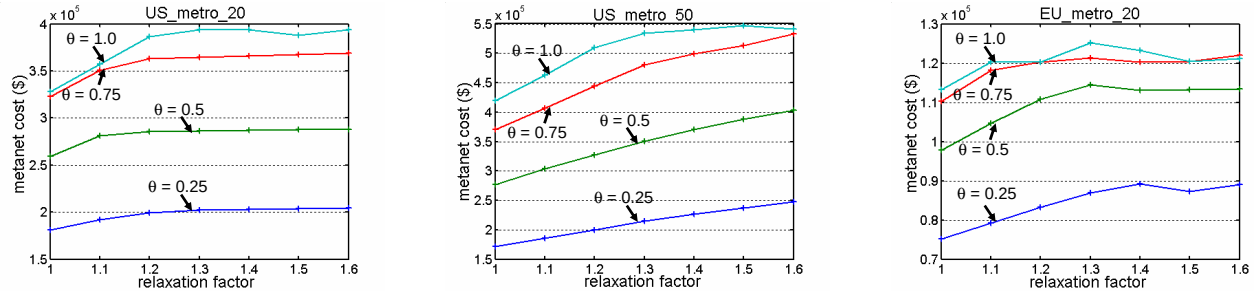
(a) US_metro_20, cost/lb = 1.38 (b) US_metro_50, cost/lb = 1.51 (c) EU_metro_20, cost/lb = 1.50

Fig. 11. The least-cost metanets on three substrates with $\delta = 1.6$ and $\theta = 0.25$



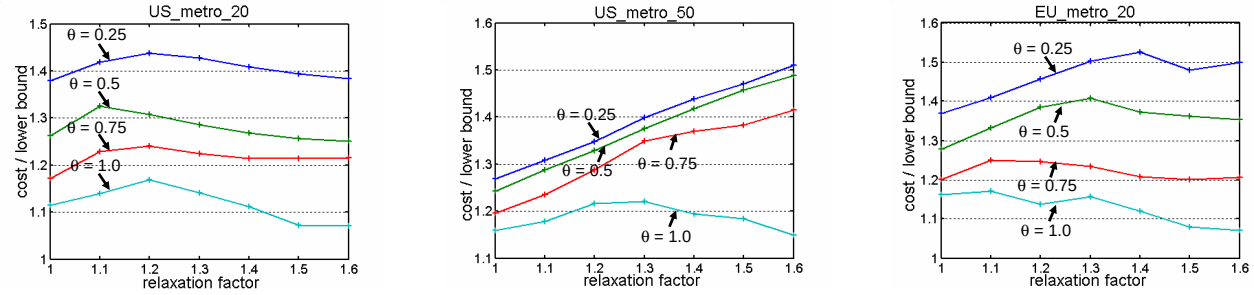
(a) US_metro_20 (b) US_metro_50 (c) EU_metro_20

Fig. 12. The lower bound on the cost of metanet configurations on the three substrates



(a) US_metro_20 (b) US_metro_50 (c) EU_metro_20

Fig. 13. The cost of the least-cost metanet configurations on the three substrates



(a) US_metro_20 (b) US_metro_50 (c) EU_metro_20

Fig. 14. The ratio of the least-cost metanet configurations to the lower bound ratio on the three substrates

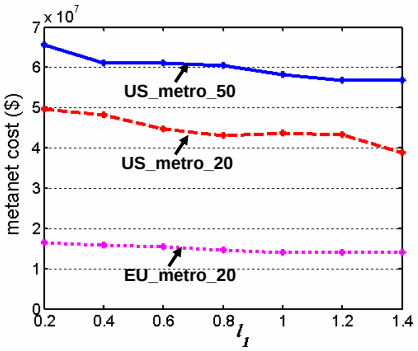


Fig. 15. Metanet cost vs. l_1 with $l_2 = 0.2$ under loose traffic constraints with $\delta = 1.6$ and $\theta = 1.0$

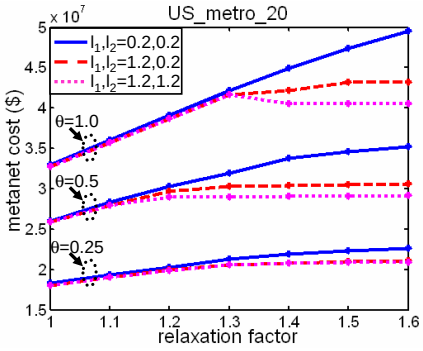


Fig. 16. Metanet costs on US_metro_20 under various traffic conditions with tight to loose delay bounds

nodes that are close on the substrate, we see the metanet costs decrease steadily for all three substrates. When l_1 is 1.4, the metanet cost is cut by 22% on US_metro_20, 16% on EU_metro_20 and 14% on US_metro_50, respectively. Clearly the results indicate that, for the metanets with loose traffic constraints, tradeoffs can be made between lowering network costs and reducing propagation delays. However, the relatively small effect also suggests that we can afford to tighten the delay bounds without having huge impact on the network cost.

Next, we study the impact of the delay bounds on the metanet costs under various traffic conditions. In particular, we select three sets of delay parameters to see to what extent, the metanet cost is affected by the delay bounds. Fig. 16 shows three groups of curves with each group associated with a distance factor. Within a group, the blue solid line indicates the metanet cost under tight delay bounds with $l_1 = 0.2$ and $l_2 = 0.2$, the red dashed line is for $l_1 = 1.2$ and $l_2 = 0.2$ which loosen the delay bounds for close-by nodes, and the pink dotted line is for $l_1 = 1.2$ and $l_2 = 1.2$ that also allow longer delays among far-away nodes. Apparently, the tightest delay

bounds always yield the most expensive configuration of the three. However, in the presence of stronger traffic constraints, the difference in costs becomes smaller or even negligible. This is because tight traffic constraints force backbone nodes to be near the access nodes so that delays become less an issue. The results for US_metro_50 and EU_metro_20 look very similar to what we see in Fig. 16. They are not shown due to the limited space.

4) Running time: Using Dinic's algorithm and the dynamic trees data structure [16] to solve maximum flow problems, we can configure a low-cost metanet with n access nodes on a substrate with m links in $O(m^3n^3\log n)$ time. Even though the time grows as the cube of m and n in the worst case, the actual running time is fairly reasonable for practical-sized substrates. We ran our experiments on a Linux 2.6.9 machine with a 2.4 GHz AMD Opteron processor. On average, it takes 1.02 seconds to configure a metanet with 20 access nodes on US_metro_20, 2.61 seconds on EU_metro_20, and 107.2 seconds to configure a metanet with 50 access nodes on US_metro_50.

VII. Closing Remarks

In this paper, we address the metanet configuration problem in a diversified internet and propose a pruning algorithm for configuring low-cost metanets on any given substrate network. The resulting metanet is guaranteed to have sufficient bandwidth to accommodate any traffic pattern allowed by user-specified traffic constraints. Through extensive studies on the metanet configuration problems with various traffic constraints and delay constraints on three different substrates, we found:

1) The system of traffic constraints has a profound influence on the least-cost metanet structure. In particular, tight pair-wise constraints favor metanet topologies in which backbone nodes are directly connected by links with just the right capacity. As constraints get looser, "tree-like" topologies with fewer backbone nodes near the center of the network are better at reducing the network costs.
2) The least-cost network structure is also affected by the underlying substrate topology. Although US_metro_50 is only a larger version of US_metro_20 with the majority of users having the same geographic distribution, the least-cost metanet topologies are quite different even when the constraints are the same.
3) The delay bounds affect the metanet costs and the impact grows when the traffic constraints get relaxed. However, the delay bounds do not affect the metanet costs in a significant way, especially when traffic constraints are strong, which indicates that we can afford tighter delay bounds without having major impact on network costs.

In contrast to other testbed configuration systems, our work starts from a higher level specification and produces a network that is dimensioned to handle the specified traffic. Traffic-based specification is more suitable for designing large-scale metanets that aim at long-term deployment. To the best of our knowledge, this work is the first one that tries to automatically determine the best network topology while considering network switching costs and propagation delays. The empirical results show that our algorithm produces metanet configurations that are demonstrably close to the lower bound and is fast enough to handle substrate networks of practical size.

We recognize that there are limitations in our work. We assume that substrate links have sufficient capacity not to constrain the configurations of the metanets. Since substrates are typically designed to have enough resources to accommodate multiple metanets, this assumption is appropriate when the substrate is not operating close to its capacity limit. However, adding substrate link capacity constraints is a natural and useful extension. One way to realize this in our algorithm is to ensure, during the pruning process, that the total provisioned bandwidth of the metalinks is no greater than the capacity of any substrate link they use. If this condition is violated, we can penalize the insufficient substrate link with a higher cost, which can force some traffic to be re-routed to other paths. We will explore this issue further in our future work.

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] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "PlanetLab: An Overlay Testbed for Broad-Coverage Services," *ACM Computer Communications Review*, vol. 33, no. 3, 2003.

[3] (2006) Global environment for network innovations. [Online]. Available: http://www.geni.net/

[4] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI Veritas: Realistic and Controlled Network Experimentation," in *ACM SIGCOMM*, 2006.

[5] J. Lu and J. Turner, "Constraint-driven Virtual Network Design on A Shared Substrate," in *Poster in 14th IEEE Internation Conference on Network Protocols*, 2006.

[6] 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.

[7] Y. Zhu and M. Ammar, "Overlay Network Assignment in PlanetLab with NetFinder," *College of Computing, Georgia Institute of Technology, Technical Report GT-CSS-06-11*, 2006.

[8] Sword. [Online]. Available: http://sword.ucsd.edu/

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

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

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

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

[13] A. Fingerhut, S. Suri, and J. Turner, "Designing Minimum Cost Nonblocking Communication Networks," in *5th International Conference on Telecommunication Systems Modelling and Analysis*, Mar. 1997.

[14] 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.

[15] J. Lu and J. Turner, "Efficient Mapping of Virtual Networks onto A Shared Substrate," *Department of Computer Science and Engineering, Washington University in St. Louis, Technical Report WUCSE-2006-35*, 2006.

[16] R. E. Tarjan, *Data Structures and Netowrk Algorithms*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1983.