Constraint Based Design of ATM Networks, an Experimental Study

Hongzhou Ma, Inderjeet Singh, Jonathan Turner

WUCS-97-15

April 97

Department of Computer Science
Campus Box 1045
Washington University
One Brookings Drive
St. Louis, MO 63130-4899

Abstract

This paper describes an experimental study of constraint-based network design. We used a novel network design tool, implemented in Java, to design representative networks joining major U.S. cities. The cost of three topologies: Best Star, Minimum Spanning Tree (MST), and Delaunay Triangulation, are compared, with and without localized traffic constraints. The best star network gives near optimal result when the traffic is only constrained by source and sink capacity of switches (flat traffic constraints). With localized traffic constraints, the most cost effective network has a structure similar to the MST. The cheapest network has a tree structure when there are only flat traffic constraints, but can have cycles when localized traffic constraints are present.

Constraint Based Design of ATM Networks, an Experimental Study

Hongzhou Ma *hma@dworkin.wustl.edu*
Inderjeet Singh *inder@dworkin.wustl.edu*
Jonathan Turner *jst@arl.wustl.edu*

1. Introduction

Integrated network technologies, such as ATM, support multimedia applications with vastly different bandwidth needs, connection request rates, and holding patterns. Because of the lack of reliable data, such as distribution of bandwidth, holding times, size and behavior of multicast calls, and the difficulty of predicting future change, the classical telecommunication network planning techniques[1] are less relevant to this new generation of networks.

A constraint-based network design model is proposed. It creates networks that can handle any combination of traffic that lies within user-specified constraints, is less sensitive to assumptions, and yields greater confidence in resulting designs. The traditional network design model, which takes traffic requirement between switch pairs as input, is a special case of the constraint-based model.

Previous research on flat traffic constraints (the only constraint is that the traffic in and out of a switch cannot exceed the sink and source capacity of that switch) [2, 4, 3], showed that when the link cost is linear to capacity, the problem of finding a minimum cost, nonblocking network is NP-Complete even for symmetric source and sink capacities. When link costs satisfy the triangle inequality, the ratio between the cost of a cheapest nonblocking star and an optimal network is at most 2 if the switch capacities are symmetric, and at most $2+\frac{\max(A,Z)}{\min(A,Z)}$ in general, where $A=\sum\alpha_i$, $Z=\sum\omega_i$.

A constraint-based network design tool has been implemented in Java, which allows network designers try out different network topologies and compare their costs to a calculated lower bound. This tool is used here to design nonblocking ATM networks for major metropolitan areas of the United States.

This paper has six sections. Section 2 introduces the constraint-based network design model. Section 3 introduces Decaf, the Java tool for constraint-based design. Section 4 characterizes the data set of metropolitan areas. Section 5 presents numerical results of nonblocking networks for three cases: the twenty largest metropolitan areas, the fifty largest metropolitan areas, and the north-eastern metropolitan areas. Section 6 gives directions for future work.

2. Constraint-Based Network Design Model

In most previous work on network design, traffic requirements are given in the form of a traffic matrix, which is a table in which each row corresponds to the traffic originating at a particular node, and each column corresponds to the traffic received at a particular node. Generating such a table can often be difficult or even impossible. For example, in an environment where traffic is primarily world wide web traffic, it is possible to specify the amount of traffic a workstation receives but it is very hard to apriori fix the source of that traffic.

Our formulation of the network design problem consists of a complete directed graph $G = (V, E)$, where each vertex represents a *switch* and each directed edge represents a *link* [3].

- Switch u has a *source* capacity $\alpha(u)$ and a *sink* capacity $\omega(u)$, representing the maximum traffic rate that can originate from or terminate at u.
- Each vertex pair (u, v) has a function $\gamma(u, v, x)$ representing the cost of constructing a link of capacity x from u to v. [1]
- The general form of a *constraint* is a switch set pair. For two sets of switches, $S = \{S_1, S_2, \ldots\}$ and $D = \{D_1, D_2, \ldots\}$, $\mu(S, D)$ is the upper bound on traffic from S to D.
- A *connection request* $R = (S, D, w)$ comprises a set of source nodes S, a set of destination nodes D and a weight w. A set of connection requests is *valid* if it does not violate the constraints. In particular, the sum of the weights of the requests containing switch u as their source and sink node, respectively, must not exceed $\alpha(u)$ and $\omega(u)$.
- A *route* T for a request R is a subgraph of G for which there is a directed path from every vetex in S to every vertex in D. A collection of routes C places a connection weight $\lambda_C(u, v)$ on an edge (u, v), which is defined as the sum of the weights of all routes that include the edge (u, v).
- A *state* of a network is a valid set of routes.
- A *routing algorithm* is a procedure that maintains a valid set of routes under the following four operations:
  1. add a new route satisfying a specified connection request;
  2. remove an existing route;
  3. add new nodes to either the source set, the destination set, or both for some route in the current state;
  4. remove nodes from either the source set, the destination set, or both for some route in the current state.
- The *reachable states* for a routing algorithm on a network with specified link capacities is the set of all states that can be reached by sequences of the four operations given above, starting from the empty state.
- A network is *nonblocking with regard to a routing algorithm A* if for every state reachable under A, no valid request can block.
- The *link cost of a network* is defined as $\sum_{(u,v)\in E} \gamma(u, v, x)$, where x is the capacity on the link. The *link cost of a state* is defined as $\sum_{(u,v)\in E} \gamma(u, v, \lambda_C(u, v))$, where $\lambda_C(u, v)$ is the connection weight placed on edge (u, v), as defined earlier.
- The cost of the most expensive valid state is a *Lower bound on the link cost of a nonblocking network.*

[1] We require that the costs satisfy the *triangle inequality*, meaning that the direct path of any given capacity between two vertices is never more expensive than an indirect path with the same capacity.

The *nonblocking network design problem* is to *determine a set of link capacities* that will yield a nonblocking network of least link cost under a specified routing algorithm. In the following discussion, we will ignore the cost of switches in the network (they can be absorbed into the cost of links). So, when we say "the cost of a network" we really mean "the link cost of the network".

There are several classes of traffic constraints that we will focus on:

flat traffic. Only source-sink constraints are present.

hierarchical clustering. Clusters of switches are defined and intra-cluster and inter-cluster traffic constraints are specified.

localized. Amount of traffic a switch sends to another switch is a (decreasing) function of the distance between the two switches.

pairwise percentage. For any two switches u and v, let $f(u,v) = \frac{\omega_v}{\sum_{w \neq u} \omega_w}$, $g(u,v) = \frac{\alpha_u}{\sum_{w \neq v} \alpha_w}$. Restrict the traffic from switch u to switch v to at most $\mu(u,v) = c * \min(f(u,v)\alpha_u, g(u,v)\omega_v)$. Here c is a constant. When $c \leq 1$, the complete graph is the cheapest. When c is large enough that $\mu(u,v) \geq \alpha_u, \omega_v$, these constraints can be neglected. c is called the relaxation factor.

Even the flat traffic constraint case is already proved to be NP-Complete [3]. Approximate algorithms can be used for designing nonblocking networks of provably small cost. It has been proved that under flat traffic constraints, when link cost is linear, the ratio between the cost of a cheapest nonblocking star and an optimal network is at most 2 if the switch capacities are symmetric ($\alpha(u) = \omega(u)$ for all u); and at most $2 + \frac{\max(A,Z)}{\min(A,Z)}$ in general, where $A = \sum \alpha_i, Z = \sum \omega_i$. [3].

Given a set of switches, with traffic constraints, the steps of network design are:

1. create links connecting all the switches.
2. select a routing algorithm to route traffic on the links under the constraints.
3. dimension the capacity of links to the maximum value of all possible combination of traffic put by the routing algorithm on the link.
4. get link cost of this network by summing up the cost of all links.
5. repeat step 1 to 4, compare the costs of different topologies, until a network meets the need is found.

The *Delaunay triangulation* [5] can be useful in the design of nonblocking networks. The Delaunay triangulation is closely related to the *Voronoi diagram.*

Let $P = \{p_1, p_2, \cdots, p_n\}$ be a set of points in the two-dimensional Euclidean plane. These are called the *sites*. Partition the plane by assigning every point in the plane to its nearest site. All those points assigned to p_i form the *Voronoi region* $V(p_i)$. $V(p_i)$ consists of all the points at least as close to p_i as to any other site:

$$V(p_i) = \{x : |p_i - x| \leq |p_j - x|, \forall j \neq i\}$$

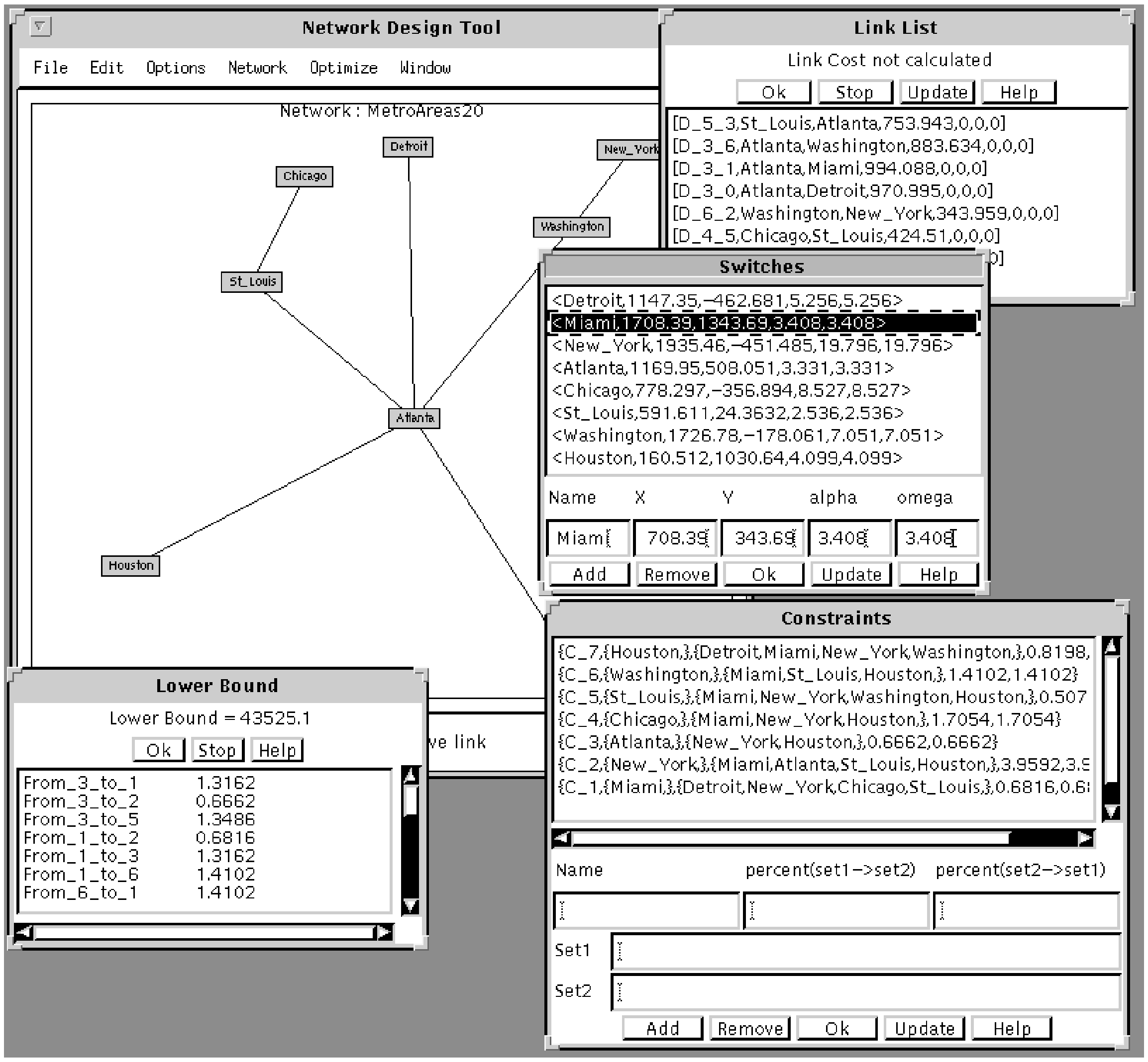
Some point do not have a unique nearest site. The set of all points that have more than one nearest neighbor form the *Voronoi diagram* $\mathrm{V}(P)$ for the set of sites.

Delaunay triangulation $\mathrm{D}(P)$ is the straight-line dual of $\mathrm{V}(P)$. The *dual* of a plane graph G assigns a node to each face and an arc for each edge between adjacent faces. The faces of $\mathrm{V}(P)$ are called *Delaunay triangles.* The bounding circle of a *Delaunay triangle* contains no other sites. A minimum spanning tree is a subset of the *Delaunay triangulation.* The *Delaunay triangulation* maximizes the smallest angle among all triangulations. In a non-strict sense, it minimizes the number of parallel links.

3. A Tool for Constraint-Based Design

A network design tool, targeted to network planners who want to deploy ATM networks, and, in the process, want to design and evaluate different network topologies, was written in Java. The initial version, called Decaf, is available at http://www.cs.wustl.edu/~javagrp/network-design-tool.html. It was implemented in JDK 1.0.2. The second version, called Cappuccino, using JDK 1.1.1, is still under development.

3.1. Functionalities of the Tool



```
Link Cost not calculated
[D_5_3,St_Louis,Atlanta,753.943,0,0,0]
[D_3_6,Atlanta,Washington,883.634,0,0,0]
[D_3_1,Atlanta,Miami,994.088,0,0,0]
[D_3_0,Atlanta,Detroit,970.995,0,0,0]
[D_6_2,Washington,New_York,343.959,0,0,0]
[D_4_5,Chicago,St_Louis,424.51,0,0,0]
```

```
<Detroit,1147.35,–462.681,5.256,5.256>
<Miami,1708.39,1343.69,3.408,3.408>
<New_York,1935.46,–451.485,19.796,19.796>
<Atlanta,1169.95,508.051,3.331,3.331>
<Chicago,778.297,–356.894,8.527,8.527>
<St_Louis,591.611,24.3632,2.536,2.536>
<Washington,1726.78,–178.061,7.051,7.051>
<Houston,160.512,1030.64,4.099,4.099>

Name     X        Y        alpha    omega
Miam     708.39   343.69   3.408    3.408
```

```
{C_7,{Houston,},{Detroit,Miami,New_York,Washington,},0.8198,
{C_6,{Washington,},{Miami,St_Louis,Houston,},1.4102,1.4102}
{C_5,{St_Louis,},{Miami,New_York,Washington,Houston,},0.507
{C_4,{Chicago,},{Miami,New_York,Houston,},1.7054,1.7054}
{C_3,{Atlanta,},{New_York,Houston,},0.6662,0.6662}
{C_2,{New_York,},{Miami,Atlanta,St_Louis,Houston,},3.9592,3.9
{C_1,{Miami,},{Detroit,New_York,Chicago,St_Louis,},0.6816,0.6
```

```
Lower Bound = 43525.1
From_3_to_1     1.3162
From_3_to_2     0.6662
From_3_to_5     1.3486
From_1_to_2     0.6816
From_1_to_3     1.3162
From_1_to_6     1.4102
From_6_to_1     1.4102
```

Figure 1: A snap shot of Decaf

In Decaf, all switches are located on a 2D plane, with x and y coordinates specified. The link cost is linear to capacity and also linear to the length. The length is calculated as the Euclidean distance between two switches, but can be overridden by the user. First, a set of links is created connecting all the switches; second, a routing algorithm is selected to route traffic on the links; last, capacities of links are calculated using linear programming, and cost is obtained.

In link creation, some useful topologies, such as best star, minimum spanning tree (MST), Delaunay triangulation [5] are provided. Best star returns a cheapest star network, so link capacities are also calculated. Delaunay triangulation is a useful topology in that it provides redundant routes between switches and is fail safe. The user can add, remove links by using the mouse. A secondary set of links can be used to adjust an initial set of links. A link in the secondary set is added to the initial graph if by adding it, the distance between two switches in the initial graph can be reduced by a pre-specified factor.

Two routing algorithms are provided: shortest path routing and distributed routing. Shortest path routing uses the shortest path between two switches on the given links. Distributed routing finds the N disjoint shortest paths between two switches on the given links; users can specify the number of different routes to use, the maximum ratio of longest route to shortest route in use, and the distribution of traffic on different routes. Shortest path routing is just a special case of distributed routing.

Link dimension and lower bound calculation are done with linear programming.

The tool can take a file as input, which specifies the switches, links, and constraints. The file format is:

```
NetworkName
⟨SwitchName, x, y, alpha, omega⟩
⋮
[LinkName, LeftSwitch, RightSwitch, length]
⋮
{ConstraintName, {SwitchSetA}, {SwitchSetB}(optional), out, in}
⋮
```

The tool also provides support for manual network creation.

3.2. Design of Decaf

Decaf was implemented in Java using JavaSoft's Java Development Kit (JDK 1.0.2). It is available both as a java applet and a java application. The tool consists of some bootstrapping code and four core packages, each of which implements a relatively independent functionality of the tool. Both applet and application share these packages but have different bootstrapping classes.

LaunchPad is the main bootstrapping class. It implements the main() method which implements the stand-alone version of the tool. LaunchPad also derives from the applet class and displays a button on the webpage which can be used to create multiple windows containing Decaf.

gui This package implements the graphical user interface of the tool. The class GUIManager is the main driver of the tool. It handles all the menu bar commands(including those for creating a network and applying algorithms to it) and manages a canvas on which the current network is displayed. Some menu options also result in the creation of pop-up dialog boxes each of which fully handles the operations defined in them. It also creates and passes an instance of a GeneralProperties data structure. It contains global information like strings for help URLs, handlers to main canvas, current network etc.

general This package contains some general data structures and utility classes. It also provides facilities for posting to and retrieving data from a CGI script. It also defines a data structure called GeneralProperties which can be used as a repository of arbitrary information indexed by strings.

network This package contains classes for representing a switch, link, network and constraints. Each of these classes implements methods for its serialization to files. The classes for dimensioning links and calculating lower bounds are derived from Thread so that these can be launched in their own threads.

lp The lp package implements a linear program solver in Java. This is a Java adaptation of an existing C program. This package is currently not being used in the tool because of anticipated slow execution of Java code. Instead, the program posts the linear programming problems to a CGI script (which uses the original C program to solve the problem) and receives the solution.

4. Characteristics of Data Sets

In 1994, the total population of the United States was some 260 million [2]. Among them, 149 million people resided in the 50 largest metropolitan areas, accounting for 57.3% of the total. So, it's of great significance to design ATM networks connecting these metro areas. Constraint based ATM network design was applied to these metropolitan areas.

Since the tool models switch locations as points in a Cartesian coordinate system, it was necessary to compute coordinates from latitude and longitude using planar projection. The results are shown in table 1.

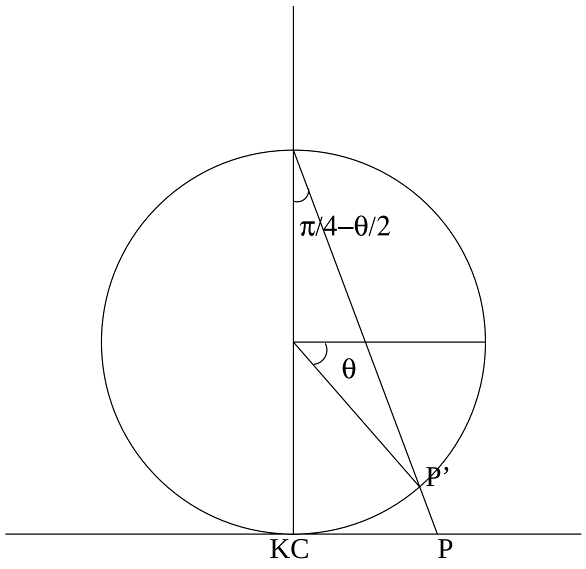


Figure 2: projection of a point from sphere to the plane

The continental part of the United States is within longitude 70° to 125°, and latitude 30° to 48°. So the point with longitude $\phi_0 = 97°$, latitude $\theta_0 = 39°$(near Kansas City) is the geographical center of the United States. Think of the Earth as a sphere, and put it on a plane, with Kansas City

[2] World Almanac 1997, p 385

Metro Areas	population(M)	latitude	longitude	x coord(Km)	y coord(Km)
New York	19.80	$40°47'$	$73°58'$	1935	-451
Los Angeles	15.30	$34°03'$	$118°15'$	-1962	332
Chicago	8.53	$41°50'$	$87°37'$	778	-357
Washington	7.05	$38°53'$	$77°02'$	1727	-178
San Francisco	6.51	$37°47'$	$122°26'$	-2232	-177
Philadelphia	5.96	$39°57'$	$75°10'$	1858	-333
Boston	5.50	$42°21'$	$71°05'$	2123	-691
Detroit	5.26	$42°20'$	$83°03'$	1147	-463
Dallas	4.36	$32°46'$	$96°46'$	22	695
Houston	4.10	$29°45'$	$95°21'$	161	1031
Miami	3.41	$25°46'$	$80°12'$	1708	1344
Atlanta	3.33	$33°45'$	$84°23'$	1170	508
Seattle	3.23	$47°37'$	$122°20'$	-1898	-1254
Cleveland	2.90	$41°28'$	$81°37'$	1282	-386
Minneapolis	2.69	$44°59'$	$93°14'$	297	-673
San Diego	2.63	$32°42'$	$117°10'$	-1894	504
St Louis	2.54	$38°35'$	$90°12'$	592	24
Phoenix	2.47	$33°29'$	$112°04'$	-1402	505
Pittsburgh	2.40	$40°27'$	$79°57'$	1442	-300
Denver	2.19	$39°45'$	$105°00'$	-684	-114
Tampa	2.16	$27°57'$	$82°27'$	1445	1132
Portland	1.98	$45°31'$	$122°41'$	-1997	-1033
Cincinnati	1.89	$39°08'$	$84°30'$	1079	-89
Kansas City	1.65	$39°06'$	$94°35'$	209	-14
Milwaukee	1.64	$43°02'$	$87°55'$	740	-488
Sacramento	1.59	$38°35'$	$121°30'$	-2126	-243
Virginia Beach	1.53	$36°51'$	$75°58'$	1871	26
Indianapolis	1.46	$39°46'$	$86°10'$	926	-141
San Antonio	1.44	$29°23'$	$98°33'$	-151	1072
Columbus	1.42	$40°00'$	$83°01'$	1191	-204
Orlando	1.36	$28°30'$	$81°22'$	1543	1054
New Orleans	1.31	$29°57'$	$90°04'$	673	986
Charlotte	1.26	$35°14'$	$80°50'$	1471	293
Buffalo	1.19	$42°55'$	$78°50'$	1479	-591
Salt Lake City	1.18	$40°46'$	$111°54'$	-1255	-302
Hartford	1.15	$41°48'$	$72°44'$	2007	-591
Providence	1.13	$41°50'$	$71°24'$	2115	-627
Greensboro	1.11	$36°05'$	$79°50'$	1544	183
Rochester	1.09	$43°10'$	$77°37'$	1571	-640
Las Vegas	1.08	$36°10'$	$115°12'$	-1635	156
Nashville	1.07	$36°10'$	$86°47'$	918	265
Memphis	1.06	$35°09'$	$90°03'$	633	406
Oklahoma	1.01	$35°26'$	$97°28'$	-42	397
Grand Rapids	0.99	$42°58'$	$85°40'$	923	-502
Louisville	0.98	$38°15'$	$85°46'$	982	23
Jacksonville	0.97	$30°22'$	$81°40'$	1481	849
Raleigh	0.97	$35°46'$	$78°39'$	1657	197
Austin	0.96	$30°16'$	$97°44'$	-71	974
Dayton	0.96	$39°47'$	$84°12'$	1094	-165
West Palm Beach	0.96	$26°42'$	$80°03'$	1707	1236

Table 1: the 21 Metro Areas in the United States

at the bottom. This will also be the origin of the plane. The new longitude and latitude values of a point on the sphere, ϕ' and θ', are related to the old values, ϕ and θ, by the following equations:

$$\begin{aligned}
\sin\phi' &= \cos\phi_0 \cos(\phi-\phi_0)\cos(\theta-\theta_0) + \sin\phi_0 \sin(\phi-\phi_0) \\
\cos\phi' \sin\theta' &= \cos(\phi-\phi_0)\sin(\theta-\theta_0) \\
\cos\phi' \cos\theta' &= \cos(\phi-\phi_0)\sin\phi_0 \cos(\theta-\theta_0) - \sin(\phi-\phi_0)\cos\phi_0
\end{aligned}$$

A point P' on the sphere is projected onto the plane by drawing a straight line from N' (point on the globe opposite to Kansas City) to P', and extending the line until it meets the plane at P. If P' has a latitude of θ, then P will be $2r\tan(\pi/4 - \theta/2)$ away from the origin.

The new latitude(θ') coordinate values of the continental part of the United States is between 70° to 90°, so the error introduced by this projection is within 1%. 6378.14Km is used as the radius of the Earth. The Euclidean distance between any two cities calculated from the X and Y coordinates is within 3% from the distance on Earth.

A single switch is located in each metropolitan area. The source and sink capacities are proportional to the population in the area, assuming that the bandwidth requirement for each person is 1Mb/s. So the switch in St. Louis would have total capacity of 2540Gb/s. It is also assumed that the link cost is $20 per kilometer per 100Mb/s. This is a reasonable assumption if 2.4Gbps links are used.

5. Numerical Results

Three cases are studied: twenty largest metropolitan areas, metropolitan areas in the north eastern part of the United States, and fifty largest metropolitan areas.

Table 2: Cost of nonblocking ATM networks for 20 largest metropolitan areas relative to lower bound (lower bound with flat traffic constraint is 64 billion dollars, with localized traffic constraints is 23 billion dollars)

	no constraint	80% traffic within 1000 Km
Best Star	1.01	2.76
Minimum Spanning Tree	1.35	1.63
Delaunay triangulation	2.55	2.56
Delaunay(distributed routing)	2.13	2.23
Hand Crafted	1.11	1.58
Hand Crafted (distributed routing)	1.13	1.61

For the twenty largest metropolitan areas, we study the impact of localized traffic constraints first. The cost of different network topologies are computed and compared, with and without localized traffic constraints. The localized traffic constraints specify that 80% of traffic in and out of a switch are limited within a circle with radius of 1000 kilometers. If the circle takes in fewer than 3 neighboring metro areas, then the radius is increased until there are exactly 3 neighboring metro areas within the circle. If the default circle takes in more than 10 neighboring metro areas, then the radius is reduced until it takes in exactly 10.

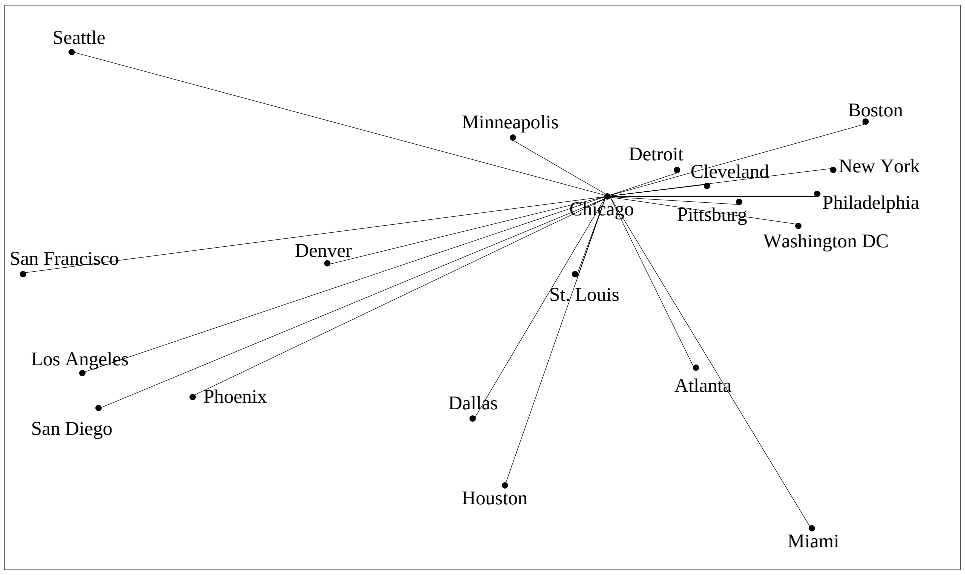


Figure 3: the Best Star Network for 20 largest metro areas

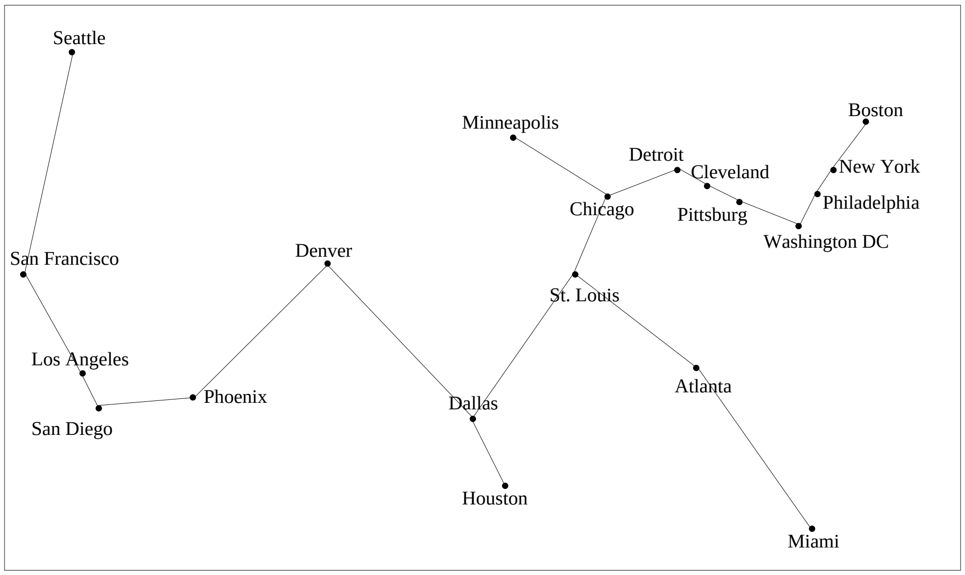


Figure 4: the Minimum Spanning Tree for 20 largest metro areas

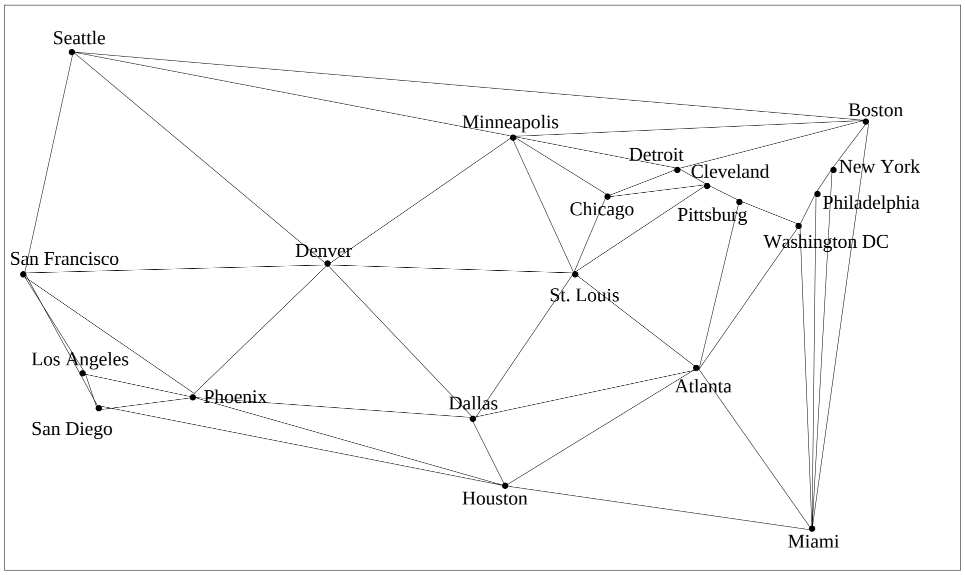


Figure 5: the Delaunay Triangulation for 20 largest metro areas

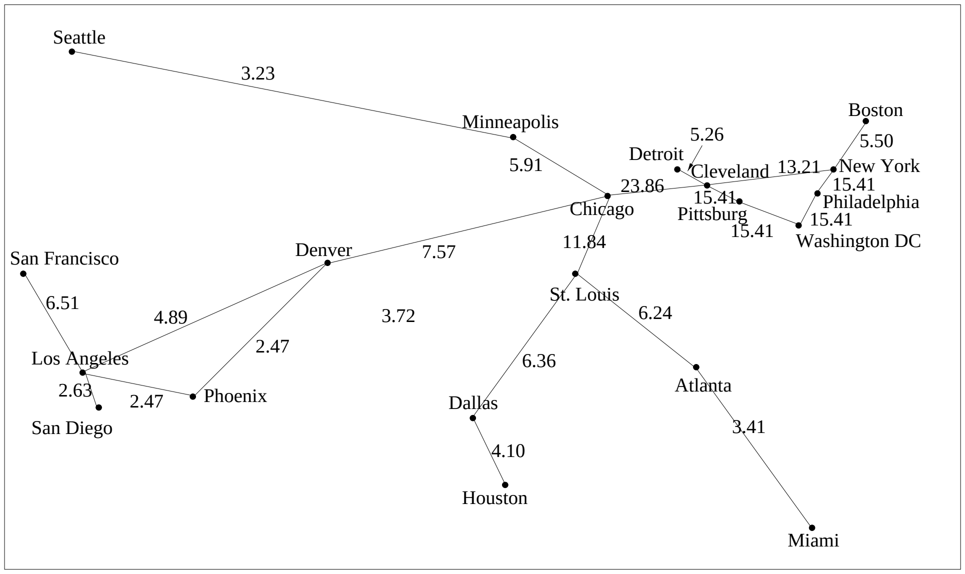


Figure 6: Hand crafted network for 20 largest metro areas with link capacities (unit: Tb/s, under distance traffic constraint)

In distributed routing for Delaunay triangulation, three shortest routes are chosen for any two switches, but any route longer than twice the length of the shortest route is discarded. The traffic distribution among the three routes are 2:1:1 when three routes are available. For example, the three shortest routes from St. Louis to Miami are St. Louis → Atlanta → Miami, St. Louis → Dallas → Houston → Miami, and St. Louis → Cleveland → Pittsburg → Washington D.C. → Miami. The length of the last one is more than twice the shortest route, so it is not used. The distribution of traffic is 2:1:1, but because the third route is not used, it's 2:1 on the two routes; 67% of the traffic goes through the shortest route, 33% of the traffic goes through the second shortest route. In distributed routing for the hand crafted network, only two shortest routes are chosen, and the traffic distribution is 2:1.

The center of the best star network for the 20 largest metro areas is Chicago, with or without localized traffic constraints. When there are only flat traffic constraints, the best star network has a cost close to the lower bound(within 1%). When the traffic is localized, the lower bound is reduced, but the cost of star network is almost unchanged, since even local traffic still has to go through the center. The cost of the star network is even more expensive than the Delaunay triangulation in this case.

The Delaunay triangulation provides alternative routes between switches, and is a useful topology for designing fail-safe cost-effective networks. Its relative cost to the lower bound is almost unchanged with or without localized traffic constraints. Distributing traffic on different routes reduces the cost.

With localized traffic, the MST is a good starting point for constructing a least cost network, because it keeps lots of short links connecting neighboring metro areas. But the MST gives every switch the same weight, no matter what their source and sink capacities are. A better solution could be obtained if links connecting major cities are given preference. In the case of the 20 largest metro areas, there is a lot of traffic goes between Los Angeles and New York, and the MST gives a zig-zag path between them. By constructing the backbone connecting Los Angeles → Denver → Chicago → Cleveland → New York, then augmenting this backbone into a tree, and adding one link in south western and north eastern areas, the cheapest nonblocking network with localized traffic constraints is obtained.

Divide and conquer is very useful in constructing a least cost network. Here, metro areas are natually divided into several regions, such as the West (west of Denver), the Northeast (east of Chicago, north of Washington DC). We can first construct least cost networks for these regions, then connect them together. When designing for the West, all cities east of Denver are remote to this region, we can put all their capacities into one single city, such as St. Louis, then connect Denver and St. Louis, and try to find the best topology connecting all the other cities. The size of the network is much smaller than the original one, and different topologies can easily be tried out.

Table 3: Cost of nonblocking ATM networks for 20 largest metro areas relative to the lower bound, with pairwise percentage traffic constraints

relaxation factor c	1	1.2	1.6	2	4	10
Lower Bound(in billion dollars)	44.8	51.4	59.2	62.0	63.1	63.8
Best Star(Chicago)	1.38	1.26	1.09	1.04	1.03	1.01
Complete Graph	1.00	1.05	1.21	1.45	2.84	5.35
Minimum Spanning Tree	1.40	1.43	1.44	1.39	1.37	1.35
Delaunay triangulation	1.05	1.10	1.25	1.43	2.06	2.39
Delaunay(distributed)	1.16	1.21	1.33	1.44	1.78	2.02
Hand Crafted	1.21	1.21	1.16	1.10	1.12	1.11

The impact of the pairwise percentage traffic constraints (as defined in Section 2) was also studied. The complete graph is the best when relaxation factor $c = 1$, but it increases linearly with relaxation factor c. The best star is the cheapest network for $c \geq 2$. The real cost of the best star network is almost not changed with different c, but the lower bound changes. Delaunay has a cost about the same as the complete graph when $c \leq 2$, and is much better than complete graph when $c \geq 4$, so it is worthy of consideration. The relative cost of the MST and the hand crafted network have the nice feature that they are almost constant with regard to c.

Table 4: Cost of nonblocking ATM networks for north eastern metropolitan areas relative to lower bound (lower bound with no constraint is 3.1 billion dollars, with distance constraint is 1.3 billion dollars)

	no constraint	80% traffic within 200 Km
Best Star(New York)	1.000	2.106
Minimum Spanning Tree	1.234	1.574
Delaunay triangulation	1.858	1.868
Delaunay(distributed routing)	2.100	1.969

For the north-eastern metropolitan areas, networks with and without localized traffic constraints are studied. Here, the radius for localized traffic constraints is taken to be 200 Km, and because there are only nine metro areas in this case, the minimum and maximum number of neighbors are taken to be 2 and 3. Here, the best network with localized traffic constraints is the MST, and the best star is centered on New York. Distributed routing for Delaunay triangulation is even more expensive than shortest path routing, suggesting that distributed routing is only good for networks with larger numbers of switches.

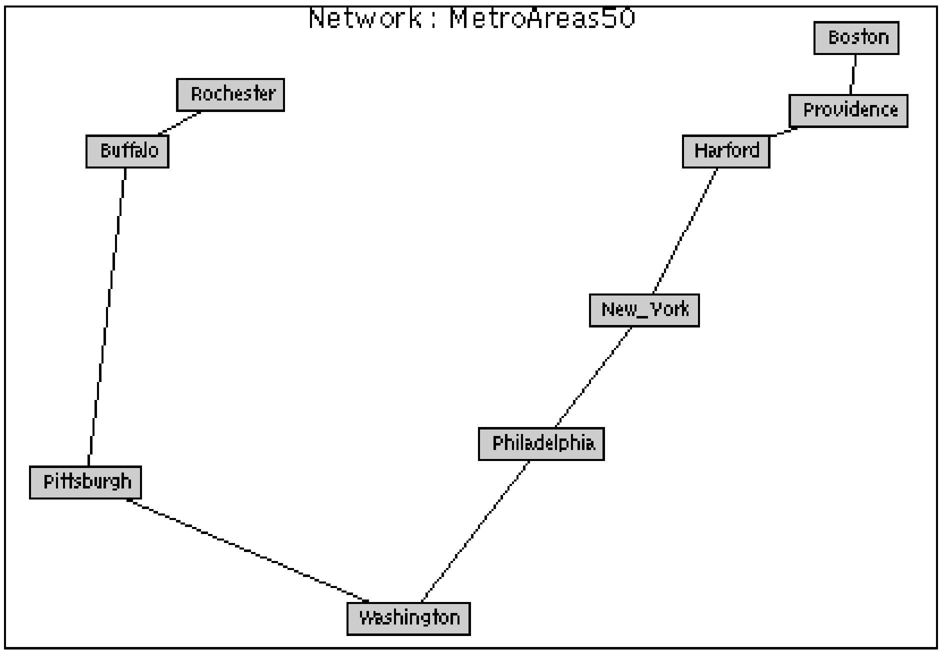


Figure 7: the Minimum Spanning Tree for north eastern metro areas

Networks for the 50 largest metro areas were also studied under the same condition as for the north-eastern areas. The center of the best star is Indianapolis, and distributed routing is much better than shortest path routing, further confirming our previous observation that distributed routing

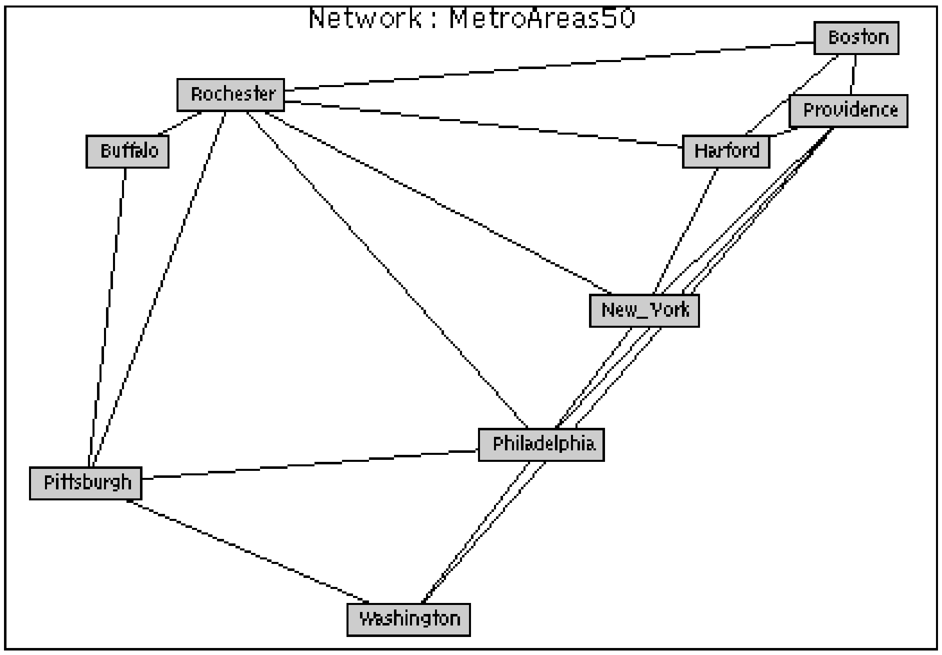


Figure 8: the Delaunay triangulation for north eastern largest metro areas

Table 5: Cost of nonblocking ATM networks for 50 largest metropolitan areas relative to lower bound (lower bound with no constraint is 81 billion dollars, with distance constraint is 27 billion dollars)

	no constraint	80% traffic within 1000 Km
Best Star(Indianapolis)	1.003	2.99
Minimum Spanning Tree	1.377	1.78
Delaunay triangulation	3.597	3.28
Delaunay(distributed routing)	2.799	2.69
Hand Crafted	1.243	1.67

is good for large networks. The hand crafted network, optimized for localized traffic constraints, has a backbone similar to the case for the 20 largest metro areas, but moved a little bit towards the south, and there is only one cycle in the northeast.

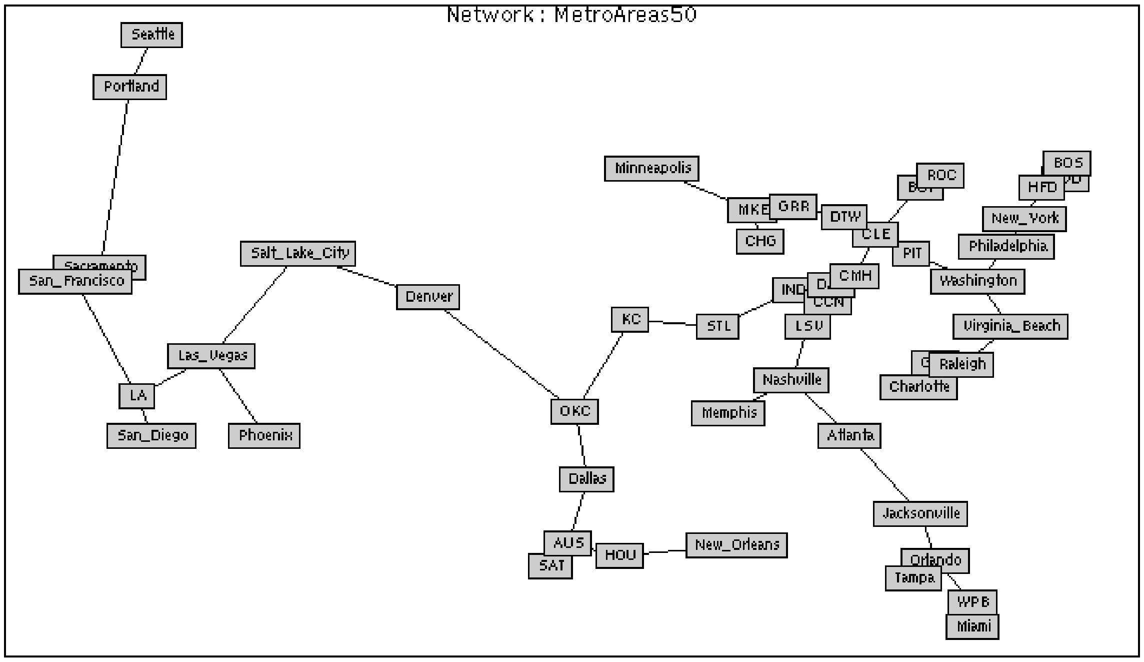


Figure 9: the Minimum Spanning Tree for 50 largest metro areas

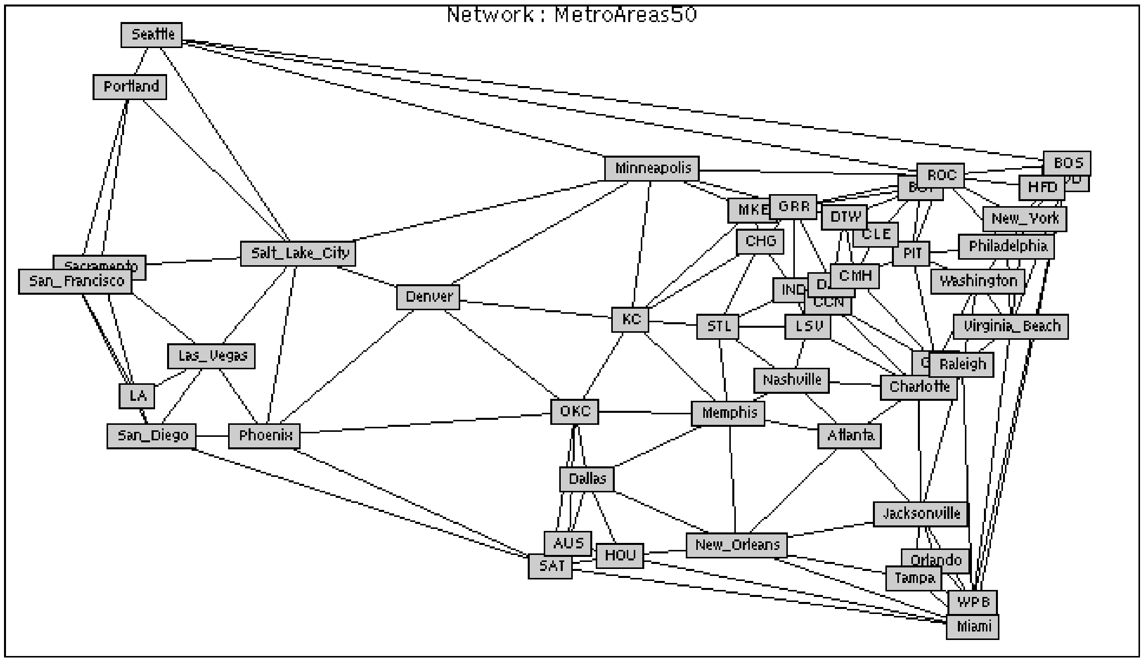


Figure 10: the Delaunay Triangulation for 50 largest metro areas

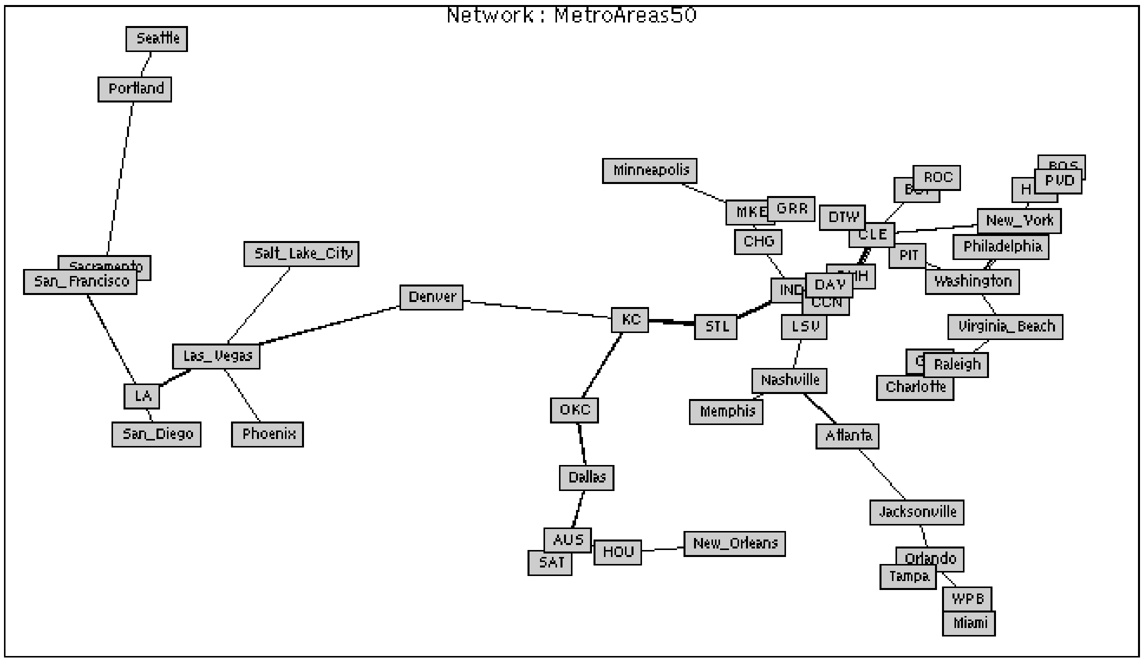


Figure 11: Hand crafted network for 50 largest metro areas

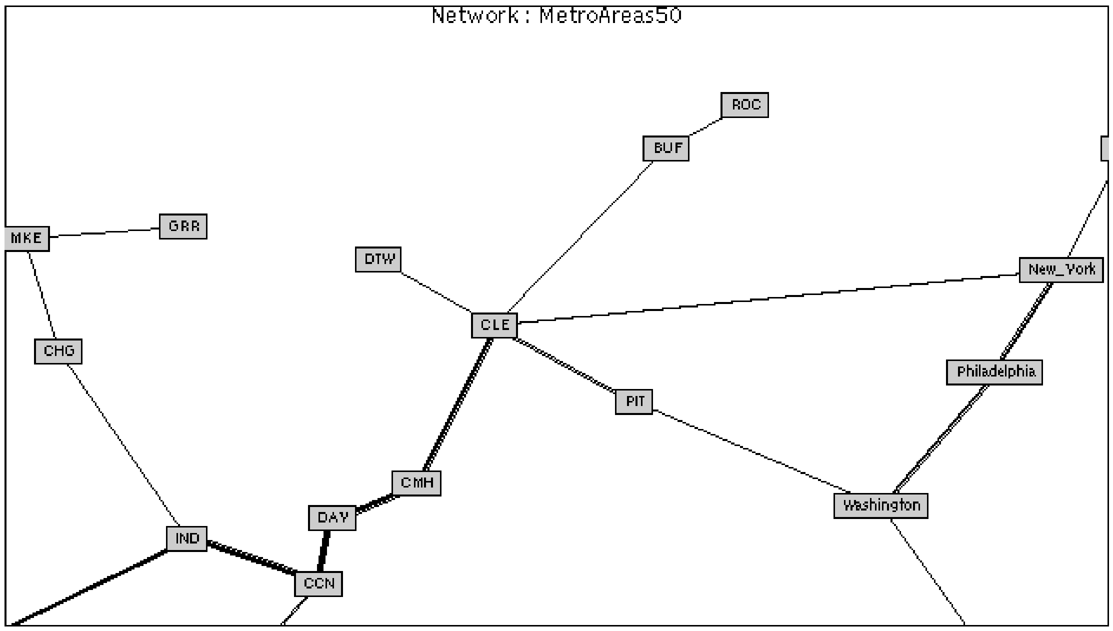


Figure 12: Hand crafted network for 50 largest metro areas (details of northeast)

Simulation shows that, if traffic is not localized, acyclic networks are always cheaper than cyclic networks. This implies that tree structure is the least expensive, and it's proved that star is best among trees [2], so the star network is almost the cheapest among all nonblocking networks. In the three cases studied, the closeness of the cost of the best star to the lower bound, also supports this observation. With local traffic constraints, cycles can help to reduce cost, as observed in the northeastern and southwestern regions in the case of twenty largest metro areas.

The experimental results suggest that different topologies are preferred, based on the relative strength of distance and pairwise constraints. This indicated qualitatively in the diagram shown in Fig. 5.

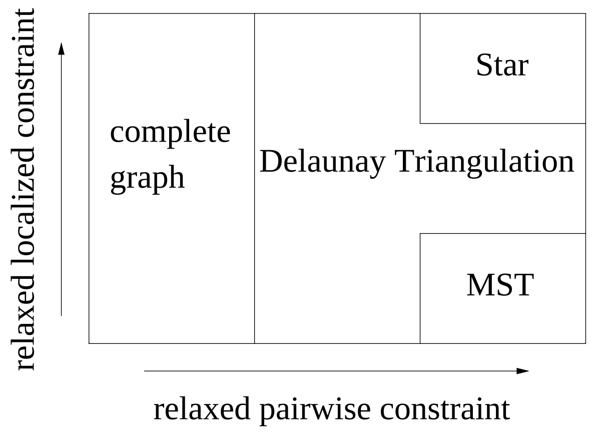


Figure 13: Topologies with the lowest cost in different situations

6. Closing Remarks

A constraint-based network design tool was implemented in Java, and used to construct ATM networks for the largest metro areas of the United States. The star network is best with flat traffic constraints. When traffic is localized, the MST is much better than the star. To design low cost networks, it is also important to minimize pathes between major nodes. To derive candidate link sets, topologies having more features, such as Euclidean spanner [6], a spanning tree that simultaneously approximates a shortest-path tree and a minimum spanning tree [7] could be helpful. Looking at the pairings that are generated by the configuration for the lower bound and create network topologies start from the pathes connecting these pairs may be another approach. The impact of other constraints, such as *hierarchical clustering*, should also be studied. Another direction for future work is make the network design tool more realistic, such as include link cost models other than linear model.

This project is built upon previous work done by J.A. Fingerhut and Subash Suri. J.A. Fingerhut contributed lots of ideas on the design of Decaf, and Subash Suri provided some of the algorithms.

References

[1] Kershenbaum, Aaron. *Telecommunications Network Design Algorithms,* McGraw-Hill, Inc. 1993.

[2] Fingerhut, J. Andrew. *Approximation Algorithms for Configuring Nonblocking Communication Networks,* Washington University Computer Science Department doctoral dissertation, 5/94.

[3] Fingerhut, J. Andrew, Subash Suri, Jon Turner. *Design Minimum Cost Nonblocking Communication Networks,* WUCS-96-06, 2/96.

[4] Fingerhut, J. Andrew, Rob Jackson, Subash Suri, Jon Turner. *Design of Nonblocking ATM Networks,* WUCS-96-03, 2/96.

[5] O'Rourke, Joseph. *Computational Geometry in C,* Cambridge University Press 1994.

[6] Arya, Sunil, Gautam Das, David M. Mount, Jeffrey S. Salowe, Michiel Smid. *Euclidean Spanners: Short, Thin, and Lanky*

[7] Khuller, Samir, Balaji Raghavachari, *Balancing Minimum Sapnning Trees and Shortest-Path Trees*