Review: MCLA Problem and NNEW Algorithm

° If a network has multiple centers, instead of building a tree, we now want to build a forest.

Definition 7.1
A forest, \( F = (V, E) \), is a simple graph without cycles
• A forest need not be connected.

° Motivation: for each center \( b \), let \( S_b \) be the set of access nodes that is closer to \( b \) than to any other center. Then, apply Esau-Williams’s algorithm on each of the sub-problems.

° NNEW Algorithm
• For each center \( b \) in its set \( B \), let \( S_b = \{ n \in N \mid \text{Cost}(n, b) < \text{Cost}(n, b') \ \forall b' \in B \} \); If \( n \) is equidistant between several backbone nodes, add \( n \) to one \( S_b \) at random.
• Use Esau-Williams to construct a capacitated MST on each set \( b \cap S_b \)
An Example of Backbone Network

- Backbone network design rule:
  - The best topology is not limited to a tree

- Example (traffic pattern):
  - Each node communicates to the 2 adjacent nodes, but not across the diagonal
  - Each piece of traffic is 32 Kbps
  - 64 Kbps available; 50% utilization constraint

Design 1: mesh
Cost: 4 64Kbps links

Design 2: tree
Cost: 3 * 2 64Kbps links

Mesh Network Design

- Backbone network design principle:
  - Direct paths for the traffic between source and destination.
  - Well-utilized components
  - Use high speed lines to achieve economy of scale
    - Cost to speed/bandwidth is not linear

- These goals are self-contradictory:
  - If all traffic direct -> a mesh of low-speed links
  - If all the components well-utilized -> trees with too many hops
  - If only have high-speed lines -> too much capacity in the periphery
Examples of Bad Design

**Too Many (slow) Direct Links; Nodes with High Degree (routing?)**

45 node network with cost=$264,411/month (180 * 1+ circuits; degree 8)
2 data centers (N1, N45), 4 servers (N2, N3, N43, N44)

**Design with Only High Speed Links**

- Another extreme case ($133,584; average # hops 7.8424)
  - Warning sign: high average number of hops (too circuitous)
2-Level Design ($96,777)

- Two-level design is natural (average # hops 3.4101)
  - 39 small nodes be connected into the network with a single 56K link
  - 2 data centers and 4 servers need more capacity
  - Pick heavy traffic nodes as interior nodes of the tree

A More Reliable Design

- Instead of tree, interior nodes form a 2-connected graph ($112,587)
  - One loop (2 F256) plus a direct link between N1 and N45 (T1)
A Different Interior Topology

- Reduce cost from $112,587 → $108,724
  - 3 T1 for N2, N45, N44, N1; 1 256K for the rest of loop

Add More Backbone Nodes

- $103,107 (cost reduced by adding N10, N13 as backbone nodes)
  - Less traffic between 2 centers N2 and N45
**Even Lower Cost Design**

- A $101,806 design
  - Backbone site selection algorithms choose N4 as the additional backbone node

**Algorithm Complexity and Design Space Size**

- Suppose the entire design space $D$ contains $10^{20}$ different designs. One random design costs 160K, and the optimal cost is 100K.
  - In average, we have to create/examine 10000 designs to get a design which costs about 125K
- Even if a subset of good designs ($D', D' \subset D$) can be identified we are still dealing with big design space
  - The huge design space push use to consider low-complexity algorithms

<table>
<thead>
<tr>
<th>Cost</th>
<th>Number of designs</th>
<th>Work in $D$</th>
<th>Work in $D'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>100,000</td>
<td>1</td>
<td>$10^{20}$</td>
<td>$5 \times 10^{16}$</td>
</tr>
<tr>
<td>101,000</td>
<td>100</td>
<td>$10^{18}$</td>
<td>$5 \times 10^{14}$</td>
</tr>
<tr>
<td>105,000</td>
<td>$10^6$</td>
<td>$10^{14}$</td>
<td>$5 \times 10^8$</td>
</tr>
<tr>
<td>110,000</td>
<td>$10^9$</td>
<td>$10^{11}$</td>
<td>$5 \times 10^5$</td>
</tr>
<tr>
<td>115,000</td>
<td>$10^{12}$</td>
<td>$10^8$</td>
<td>$5 \times 10^3$</td>
</tr>
<tr>
<td>125,000</td>
<td>$10^{16}$</td>
<td>$10^4$</td>
<td>5</td>
</tr>
<tr>
<td><strong>160,000</strong></td>
<td>$10^{28}$</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Network routing and backbone design

**Why routing is an issue in Mesh networks (but not in Trees)?**

- **Example**
  - If each link has a capacity of 10
    - a simple minimum hop routing works
  - If each link has a capacity of 6
    - Route AH via E, and ED via A => a flow of 5 on AE; or minimum hop?
  - If each link has a capacity of 5
    - Above routing results in 100% of AE (infeasible for PS)
    - Alternative: divide AH flow among 2 routes (2 via B, 2 via E); ED via F?
  - How about each link has a capacity of 3? No way!

\[
\text{Traffic} = \{ \text{AH = 4, ED = 1, BC = 1, CD = 1} \}
\]

**Mentor Algorithm [KKG91]**

Assume single link type with capacity \( C \).

1. Choose backbone sites. (Also called Threshold Clustering Algorithm)
   - Calculate the normalized weight \( \text{NW}(\text{Ni}) = \frac{\text{Weight}(\text{Ni})}{C} \)
   - Choose sites with \( \text{NW}(\text{Ni}) > \text{WPARM} \) (threshold)
   - Group end sites around a backbone site, \( E \), based on
     \[ \text{Cost}(E, \text{Ni})/\text{MAXCOST} < \text{RPARM}. \]
     Where \( \text{MAXCOST} = \max_{i,j} \text{Cost}(\text{Ni}, \text{Nj}) \)
   - If there are sites not covered in groups, compute
     \( \text{merit}(n) = 1/2 \times (\text{MaxDistCtr}-\text{distCtn})/\text{MaxDistCtr} + 1/2 \times (\text{Weightn}/\text{WeightMax}) \)
     Here
     \[
     \text{DistCtn} = \sum_{n,i} (x_n - x_{ctr})^2 + (y_n - y_{ctr})^2
     \]
     \[
     \text{x}_{ctr} = \frac{\sum_{n,i} x_n \times \text{Weight}_n}{\sum_{n,i} \text{Weight}_n}
     \]
     \[
     \text{y}_{ctr} = \frac{\sum_{n,i} y_n \times \text{Weight}_n}{\sum_{n,i} \text{Weight}_n}
     \]
     Center of Mass \((x_{ctr}, y_{ctr})\) defined by
     \[
     \text{MaxDistCtr} = \max_{\text{Ni}, \text{Nj}} \sqrt{(x_{Ni} - x_{ctr})^2 + (y_{Ni} - y_{ctr})^2}
     \]
     the merit function gives equal value to the node’s proximity to the center and its weight
   - Sort the merit functions. The node with largest merit get picked as backbone node. Group end node around it. Repeat until all nodes are either backbone nodes or covered in groups of a backbone node
Mid Stage of Threshold Cluster Algorithm

° Big Squares are Backbone nodes
° The small squares in the circles are sites picked as end sites
  • Radius of each circle = RPARM * MAXCOST

Final Stage of Threshold Clustering

° Based on merit(), three backbone nodes are picked.
Mentor Algorithm Steps 2-3

2. Pick median node (root node of the whole network) with smallest Moment():
   \[ \text{Moment}(n) = \sum_{n' \in N} \text{dist}(n, n') \times \text{Weight}_n \]

3. Build a restricted Prim-Dijkstra tree rooted at median. Here only backbone nodes can be the interior nodes of the tree.

4. Sequencing Node Pair: Prepare adding additional direct links to the tree.
   - Use the tree to list node pair in “sequence”
     The node pair with longer path will list first
   - Choose home node H for each node pair (Ni, Nj) (H and Nx are intermediate nodes along the path) that satisfies
     \[ \text{Cost}(Ni, H) + \text{Cost}(H, Nj) \leq \text{Cost}(Ni, Nx) + \text{Cost}(Nx, Nj). \]

5. Adding direct links if sufficient traffic

3rd: Restricted Prim-Dijkstra Tree

- Note that there is an end node that violate the constraint
  - Here only backbone nodes can be the interior nodes of the tree
  - Solution: one additional short link
4th(a): Sequencing Node Pairs

- Sequencing the pairs in decreasing order of # hops in the tree
  - Not unique sequence, but single direction

4th(b): homing

For each pair that is not adjacent in the tree, select a home node

- Choose home node \( H \) for each node pair \( (N_i, N_j) \) (\( H \) and \( N_x \) are intermediate nodes along the path) that satisfies
  \[
  \text{Cost}(N_i, H) + \text{Cost}(H, N_j) \leq \text{Cost}(N_i, N_x) + \text{Cost}(N_x, N_j).
  \]

Example

- If \( N_1 \) and \( N_2 \) are 2 hops apart, the home is the unique node between them
- If \( N_1 \) and \( N_2 \) are more than 2 hops apart, there are 2 candidates for the home
  - Let \( N_3 \) be the first node on the path from \( N_1 \) to \( N_2 \)
  - Let \( N_4 \) be the first node on the path from \( N_2 \) to \( N_1 \)
  - If \( \text{Cost}(N_1, N_3) + \text{Cost}(N_3, N_2) \leq \text{Cost}(N_1, N_4) + \text{Cost}(N_4, N_2) \)
    - \( N_3 \) is the home \( H \)
    - Otherwise, \( N_4 \) is the home \( H \)
5th Step of Mentor Algorithm: adding direct links

5. Decide which node pairs deserve direct links
   • Assumption: symmetric traffic matrix
   • Start with the top node pair (N1,N2) in the sequence
   • Calculate the utilization $u = \frac{\text{Traf}(N1,N2)}{n \times C}$
     where $n = \text{ceil}(\text{Traf}(N1,N2)/C)$
   • If $u > \text{util}_{\text{min}}$, add direct link between $N1$ and $N2$
   • If $u < \text{util}_{\text{min}}$, add $\text{Traf}(N1,N2)$ to $\text{Traf}(N1,H)$ and $\text{Traf}(H,N2)$
     - Traffic addition for both directions
     - Here $H$ is the home of $(N1,N2)$
   • Remove $(N1,N2)$ from the sequence and repeat Step 5 again until all node pairs are processed
   • Motivation: aggregate traffic that can be used to justify links that connect sites that are several hops apart in the tree. If the traffic between two nodes can't justify a link, it is detoured through the home node.

Complexity of Mentor Algorithm

° The three basic steps: backbone selection, tree building, and direct link addition are all $O(n^2)$.
° It can be executed pretty fast due to low complexity
° Typically we will generate a set of designs based on the different threshold parameters, e.g.,
  • different WPARM, RPARM,
  • Difference $\alpha$ in the restricted Prim-Dijkstra tree, or
  • different node pair sequence (note that the sequence are not unique)
° We then pick the best design from the set.
Example of Mentor Algorithm Result

- 15 sites, 5 backbone nodes, total traffic 60 256Kbps, $269K design

Principle: if a design has high-speed links configured with several circuits in parallel, there is almost always a meshier network with lower cost and greater routing diversity

Mentor Algorithm Design 2

- $221,590, same 5 backbone nodes, with lower util_{min} = 0.7
  - Average # hops reduced from 2.850 to 2.117
Mentor Algorithm Design 3

- Mentor algorithm is different to EW or MSLA: produce suites of designs rather than a single isolated design due to the low algorithm complexity.
- Same 5 backbone nodes but with different tree. $209,220.

Cost of Designs vs. $\alpha$ and $\text{util}_{\text{min}}$

- $\alpha = 0.1$ and $1 - \text{util}_{\text{min}} = 0.1$ is the best value.
Cost vs. Size of Backbone

- The lowest-cost Mentor network for different sizes of backbones

<table>
<thead>
<tr>
<th>Backbone sites</th>
<th>Best cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>209,220</td>
</tr>
<tr>
<td>6</td>
<td>203,445</td>
</tr>
<tr>
<td>7</td>
<td>206,650</td>
</tr>
<tr>
<td>8</td>
<td>205,320</td>
</tr>
<tr>
<td>9</td>
<td>200,055</td>
</tr>
<tr>
<td>10</td>
<td>201,025</td>
</tr>
<tr>
<td>11</td>
<td>201,025</td>
</tr>
<tr>
<td>12</td>
<td>198,975</td>
</tr>
<tr>
<td>13</td>
<td>191,395</td>
</tr>
</tbody>
</table>

Network Loaders and Minimum Hop Routing

- Network loader: a piece of code that takes a design and loads the traffic according to the routing algorithms
- SRMH: a single route, minimum hop loader computes a single, minimum hop path between each site pair and uses that to load the traffic. If traffic doesn’t load on the fixed route, it is simply blocked.

Mentor added the link to carry the traffic from B to \{F, G, H, I\}, but not traffic from F to \{A, B, C\}.

What SRMH routing will do?
An infeasible design with SRMH routing

- 5-backbone design costs $209,220; 100% traffic loadable
- The 12-backbone design costs $198,975; new links attract too much traffic, only 75% traffic could be loaded

Feasibility of designs with SRMH routing

- SRMH is enemy of our designers
  - adds additional constraints that the network must work even if the router chooses the worst possible minimum hop path

<table>
<thead>
<tr>
<th>Backbone sites</th>
<th>Cost</th>
<th>SRMH loading</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>209,220</td>
<td>100%</td>
</tr>
<tr>
<td>6</td>
<td>203,445</td>
<td>100%</td>
</tr>
<tr>
<td>7</td>
<td>206,650</td>
<td>98.3%</td>
</tr>
<tr>
<td>8</td>
<td>205,320</td>
<td>98.3%</td>
</tr>
<tr>
<td>9</td>
<td>200,055</td>
<td>91.7%</td>
</tr>
<tr>
<td>10</td>
<td>201,025</td>
<td>91.7%</td>
</tr>
<tr>
<td>11</td>
<td>201,025</td>
<td>91.7%</td>
</tr>
<tr>
<td>12</td>
<td>198,987</td>
<td>95.0%</td>
</tr>
<tr>
<td>13</td>
<td>191,395</td>
<td>90.0%</td>
</tr>
</tbody>
</table>

- Alternative: a network topology where only 1 minimum hop route for all traffic
  - Using 6 sites, produce a non-tree topology that have the property that all the minimum hop paths between all the site pairs are unique
A flow-sensitive, minimum hop (FSMH) routing

- FSMH: takes a requirement and loads it on a minimum hop path using only links with enough capacity to carry the traffic. The capacity of the links is computed as the difference between the true capacity and the flow currently on the link. If no path exists, the traffic is blocked.

- An example when FSMH outperforms SRMH

Each link capacity 2;
Traffic:
- A -> D  2
- A -> E  1
- A -> B  1
- B -> C  1
- C -> E  1

- Why? -- because FSMH tries harder than SRMH; a greedy loader

FSMH better than SRMH?

- Is FSMH always better than SRMH?
  - No guarantee; the performance depends on the order of the traffic loading

Traffic:
{AB, AB, AC, CD, BD}
Feasibility of designs: SRMH vs. FSMH

* When the optimization begins to find networks costing under 200K

<table>
<thead>
<tr>
<th>Backbone sites</th>
<th>Cost</th>
<th>SRMH</th>
<th>FSMH</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>209,220</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>6</td>
<td>203,445</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>7</td>
<td>206,650</td>
<td>98.3%</td>
<td>100%</td>
</tr>
<tr>
<td>8</td>
<td>205,320</td>
<td>98.3%</td>
<td>100%</td>
</tr>
<tr>
<td>9</td>
<td>200,055</td>
<td>91.7%</td>
<td>100%</td>
</tr>
<tr>
<td>10</td>
<td>201,025</td>
<td>91.7%</td>
<td>100%</td>
</tr>
<tr>
<td>11</td>
<td>201,025</td>
<td>91.7%</td>
<td>100%</td>
</tr>
<tr>
<td>12</td>
<td>198,987</td>
<td>95.0%</td>
<td>93.3%</td>
</tr>
<tr>
<td>13</td>
<td>191,395</td>
<td>90.0%</td>
<td>96.7%</td>
</tr>
</tbody>
</table>

A single-route, minimum distance (SRMD) routing

* Open Shortest Path Routing (OSPF)
  * The most commonly used routing in IP
  * Traffic is routed by a shortest-distance algorithm

* The question is: how do we design for minimum distance networks

* SRMD: uses lengths (instead of the number of hops) assigned to the links to produce routes. It computes a single, minimum distance path between each site pair and uses that to load the traffic
  * If multiple paths exist, it is assumed that the loader is free to pick any path it chooses
  * If traffic saturates the fixed route, it is discarded
  * We assume the discard mechanism is out the control of the network designer (instead, to routing designer)
**A single-route, minimum distance (SRMD) routing**

- How to set the link lengths correctly for OSPF routing?

  - The network designer added link (A, H) for carrying the traffic between A to H and B to H, not traffic of A to G, and others
    - By SRMD: A-to-G traffic take the 5-hop path of length 490, instead of the 2-hop path of length 495
    - What SRMH will do?

- The table shows the link lengths:

<table>
<thead>
<tr>
<th>End 1</th>
<th>End 2</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>90</td>
</tr>
<tr>
<td>B</td>
<td>D</td>
<td>100</td>
</tr>
<tr>
<td>D</td>
<td>E</td>
<td>100</td>
</tr>
<tr>
<td>E</td>
<td>F</td>
<td>100</td>
</tr>
<tr>
<td>F</td>
<td>G</td>
<td>100</td>
</tr>
<tr>
<td>G</td>
<td>H</td>
<td>100</td>
</tr>
<tr>
<td>A</td>
<td>H</td>
<td>395</td>
</tr>
</tbody>
</table>

**Mentor-I: one modification**

- Step 3: building Prim-Dijkstra tree
  - calculate hops[n1][n2] for the number of tree edges on the unique path in the tree from n1 to n2
  - When we put a direct edge between 2 nodes, we want to carry all the direct traffic rather than having the traffic flow through the tree
    - Hop-based distance weighing
      - Length(tree-edge) = 100
      - Length(direct_edge) = 100 + 90 * (hops[n1][n2] – 1)

  - Example:
    - Length (A, H) = 100 + 90 * (6 - 1) = 550
    - For A-to-H traffic:
      1) 6-hop path of length 600
      2) 1-hop of length 550
    - For B-to-H traffic:
      1) 5-hop path of length 500
      2) 2-hop path of length 640
Design A by Mentor-I with SRMH routing

- Total traffic is $T = 8C$; 2 links available for designs (C, 32C)
- Mentor-I: a Prim-Dijkstra tree and 1 additional link; 100% loadable

Design B by Mentor-I with SRMH routing

- Total traffic is $T = 24C$; 2 links available for designs (C, 32C)
- Mentor-I: a highly meshed backbone network
  - If hop-based distance weighting and the routing is infeasible
**Balancing for Mentor-I**

- We can further change Mentor-I by
  1) Lengthen the length of overloaded links?
  2) Shorten underutilized links?
  3) And more subtle measures?

- Or, a new version of Mentor? Mentor-II

---

**Review: Mid Stage of Threshold Cluster Algorithm**

- Big Squares are Backbone nodes
- The small squares in the circles are sites picked as end sites
  - Radius of each circle = RPARM * MAXCOST

Diagram:

- Big Squares are Backbone nodes
- The small squares in the circles are sites picked as end sites.
  - Radius of each circle = RPARM * MAXCOST
Drawbacks of Threshold Clustering

- It does not do a good job of selecting backbone nodes when all the nodes have about the same traffic
- It is insensitive to sites with low cost
  - An expensive site may be selected due to higher traffic, e.g., 5 times more expensive with twice traffic

Threshold Clustering with Pre-selected Types

- Motivation: allow the clustering algorithm less scope
- An example of SITES table

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>VCORD</th>
<th>HCORD</th>
<th>POPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATL</td>
<td>ALG</td>
<td>07260</td>
<td>02083</td>
<td>39</td>
</tr>
<tr>
<td>BAL</td>
<td>ALG</td>
<td>......</td>
<td>......</td>
<td>10</td>
</tr>
<tr>
<td>CHI</td>
<td>B</td>
<td>......</td>
<td>......</td>
<td>110</td>
</tr>
<tr>
<td>BUF</td>
<td>E</td>
<td>......</td>
<td>......</td>
<td>3</td>
</tr>
<tr>
<td>DEN</td>
<td>ALG</td>
<td>......</td>
<td>......</td>
<td>25</td>
</tr>
<tr>
<td>LAX</td>
<td>B</td>
<td>......</td>
<td>......</td>
<td>150</td>
</tr>
</tbody>
</table>

* assumption: traffic is proportional to population
K-Means Clustering

Motivation:
- Come from the arena of speech encoding
- “Toll quality” speech, transferring 8000 8-bit samples per second
- If we want to compress the speech so as to save the bandwidth consumption
  - Sending fewer samples
  - Transmitting each sample in fewer bits
  - Or both
    - In video streaming: frame rate, frame size, color depth
    » Alternative: streaming bit rate
- Question: if we decide to use 4-bit samples (16 values to transfer), what are the best 16 values to use?

K-Means Clustering (cont.)

Choose K backbone sites.
- Let S be the set of sites. We assign each site si coordinates (Xi, Yi)
- Initially choose K (e.g., K=16) random points, c1, c2, …, C16, as centers
- For each center ci, compute
  \[ S_i = \{ s \in S \mid \text{dist}(s, c_i) \leq \text{dist}(s, c_j), \quad j \neq i \} \]
- Now compute \( c_i' = \text{center}(S_i) \). Here
  \[
  \text{center}(S_i) = (x_{ctr}, y_{ctr})
  \]
  \[
  x_{ctr} = \frac{\sum_{s \in S_i} x_s \times \text{Weight}_s}{\sum_{s \in S_i} \text{Weight}_s}
  \]
  \[
  y_{ctr} = \frac{\sum_{s \in S_i} y_s \times \text{Weight}_s}{\sum_{s \in S_i} \text{Weight}_s}
  \]
- If \( c_i' \) have converged to \( c_i \), stop. Otherwise, iterate back by letting \( c_i = c_i' \), and going back to the computation of \( S_i \).
- When the \( c_i \) have converged, pick the site in \( S_i \) closest to \( c_i \) and make it the backbone site. Make all the other sites in \( S_i \) end sites.
- Question: how to decide when the algorithm has converged?
K-Means Clustering (cont.)

- Key advantage of K-means clustering over threshold clustering
  - It makes possible for the designer to carefully control the number of backbone locations, which is more difficult with the threshold clustering.

How large should the backbone be?

- A 2-means clustering design.
A 15-means clustering costs $87,440, and increases #hops from 3.0988 to 3.7445