Compact Oblivious Routing

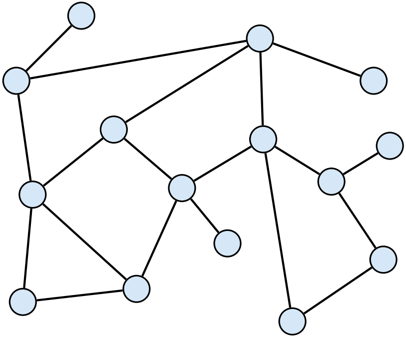
Harald Räcke, Stefan Schmid

Fakultät für Informatik
TU München

Routing in Networks

Input:

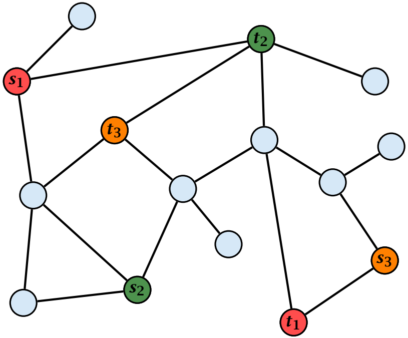
- undirected network
  $G = (V, E)$



Routing in Networks

Input:

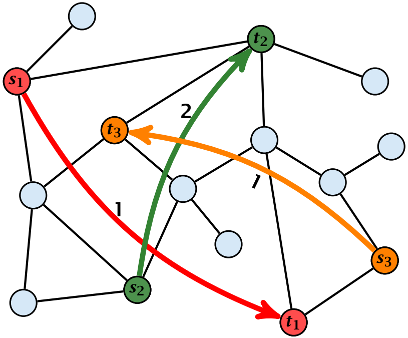
- undirected network
  $G = (V, E)$
- source/target pairs
  (s_i, t_i)



Routing in Networks

Input:

- undirected network $G = (V, E)$
- source/target pairs (s_i, t_i)
- demand d_i for i-th pair



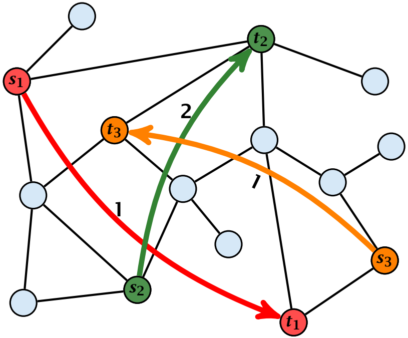
Routing in Networks

Input:

- undirected network $G = (V, E)$
- source/target pairs (s_i, t_i)
- demand d_i for i-th pair

Output:

- flow of value d_i for every pair



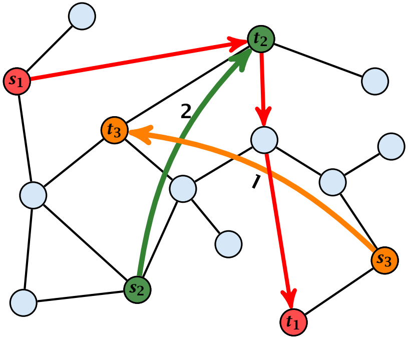
Routing in Networks

Input:

- undirected network $G = (V, E)$
- source/target pairs (s_i, t_i)
- demand d_i for i-th pair

Output:

- flow of value d_i for every pair



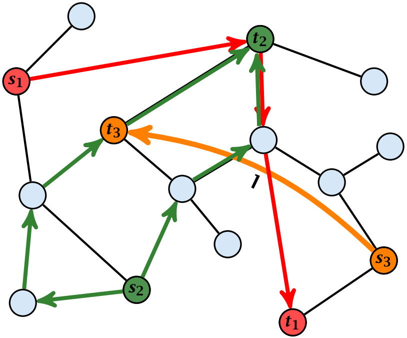
Routing in Networks

Input:

- undirected network $G = (V, E)$
- source/target pairs (s_i, t_i)
- demand d_i for i-th pair

Output:

- flow of value d_i for every pair



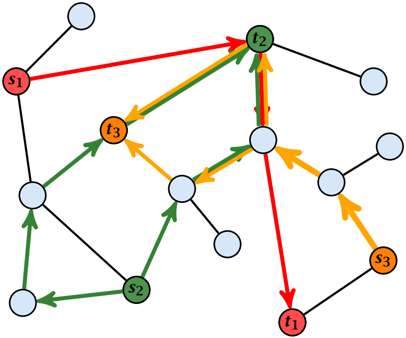
Routing in Networks

Input:

- undirected network $G = (V, E)$
- source/target pairs (s_i, t_i)
- demand d_i for i-th pair

Output:

- flow of value d_i for every pair



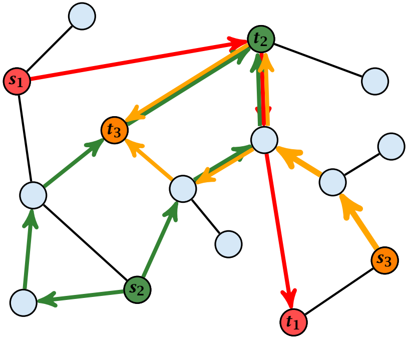
Routing in Networks

Input:

- undirected network $G = (V, E)$
- source/target pairs (s_i, t_i)
- demand d_i for i-th pair

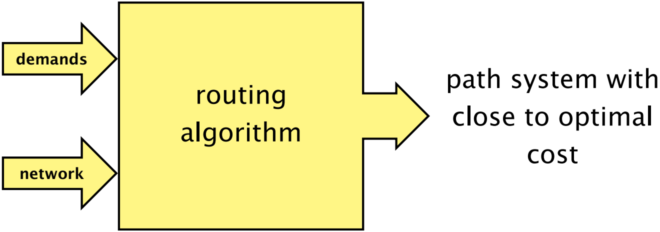
Output:

- flow of value d_i for every pair



Oblivious Routing

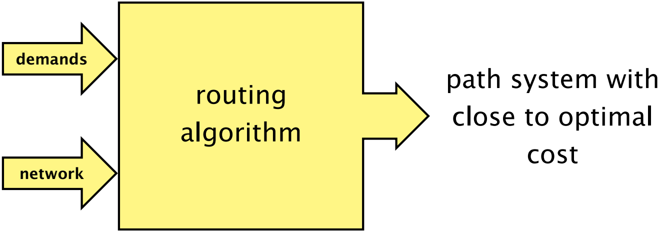
- optimization problem:



- difficult to implement in a distributed fashion
- ideally paths should be independent of demands

Oblivious Routing

- optimization problem:

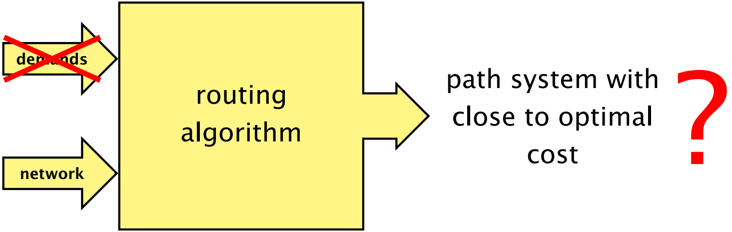


- difficult to implement in a distributed fashion

- ideally paths should be independent of demands

Oblivious Routing

- optimization problem:



- difficult to implement in a distributed fashion
- ideally paths should be independent of demands

Oblivious Routing

Oblivious Routing Scheme:

- specifies unit flow for every source/target pair without knowing any demands
- when demand d_i appears the unit flow between s_i and t_i is scaled by demand

very natural concept

Oblivious Routing

Competitive ratio of algorithm A:

$$\max_{\text{demand } d}\left\{\frac{\text{cost}(A,d)}{\text{opt}(d)}\right\}$$

Cost Measures

add explanation for length and capacity

What do we want to optimize?

load

- total traffic in the network
- $\sum_e \ell(e) \cdot \mathrm{flow}(e)$

congestion

- maximum traffic along a network link
- $\max_e \{\mathrm{flow}(e)/c(e)\}$

Oblivious Routing

Upper Bounds

load

- Shortest Path Routing is oblivious
- $\Longrightarrow$ competitive ratio: 1

congestion (undirected graphs)

- **[R. 2002]**
  competitive ratio: $\mathcal{O}(\log^3 n)$
- **[Harrelson, Hildrum, Rao 2003]**
  competitive ratio: $\mathcal{O}(\log^2 n \log\log n)$
- **[R. 2008]**
  competitive ratio: $\mathcal{O}(\log n)$

Oblivious Routing

Lower Bounds

congestion

- **[Bartal, Leonardi 1997]**
  competitive ratio: $\Omega(\log n)$ on undirected graphs
- **[Ene, Miller, Pachocki, Sidford, 2016]**
  competitive ratio: $\Omega(n)$ for directed graphs

Compact Oblivious Routing

Try to implement path selection scheme with small routing tables.

Two variants:
A packet enters the network at the source with the unique name of the destination.

labeled
the designer of the routing scheme can assign names to the vertices of the network

name-independent
the node names are fixed and cannot be changed

Compact Routing for Load

Extensively analyzed!

Parameters:

space: size of largest routing table at a node in the network

stretch: the competitive ratio w.r.t. cost-measure load

Additional parameters: header-size, label-size.

Compact Routing for Load – Results

[Folklore]
stretch: 1, space: $\mathcal{O}(n \log n)$.

[Thorup, Zwick 2001]
stretch: $4k-5$, space: $\tilde{\mathcal{O}}(n^{1/k})$, labelled

[Abraham, Gavoille, Malkhi, 2006]
stretch: $\mathcal{O}(k)$, space: $\tilde{\mathcal{O}}(n^{1/k})$, name-independent

Compact Routing for Congestion

No results!

Parameters:

space: size of largest routing table in the network; goal: $\mathcal{O}(\alpha(n) \cdot \deg(v))$, i.e., we assume space at nodes grows proportional to degree.

quality: the competitive ratio w.r.t. congestion goal: $\mathcal{O}(\operatorname{polylog} n)$

label-size: the size of assigned labels goal: $\mathcal{O}(\operatorname{polylog} n)$

header-size: the size of routing headers goal: $\mathcal{O}(\operatorname{polylog} n)$

Compact Routing for Congestion

Mark the scheme that we are using.

Oblivious routing schemes with good competitive ratio:

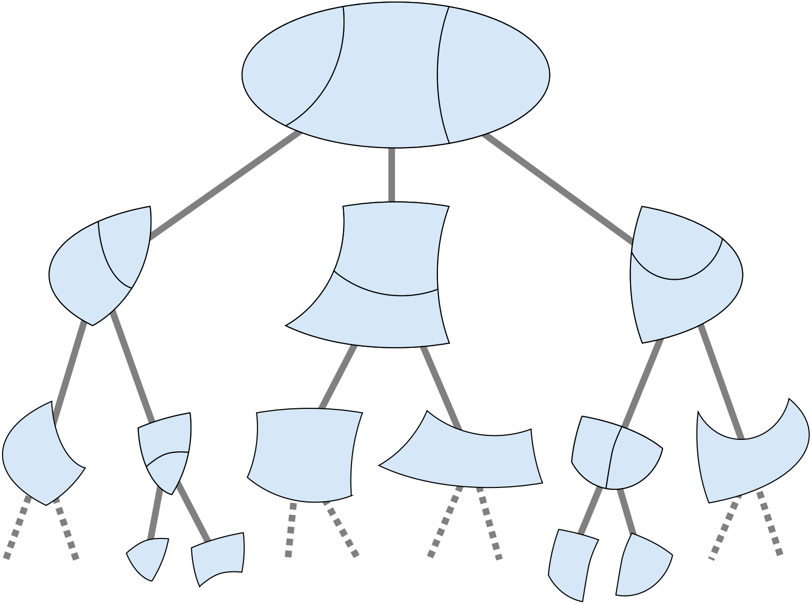
based on hierarchical decomposition

- **[R. 2002]**
  competitive ratio: $\mathcal{O}(\log^3 n)$
- **[Harrelson, Hildrum, Rao 2003]**
  competitive ratio: $\mathcal{O}(\log^2 n \log\log n)$
- **[R., Shah, Täubig 2014]**
  competitive ratio: $\mathcal{O}(\log^4 n)$

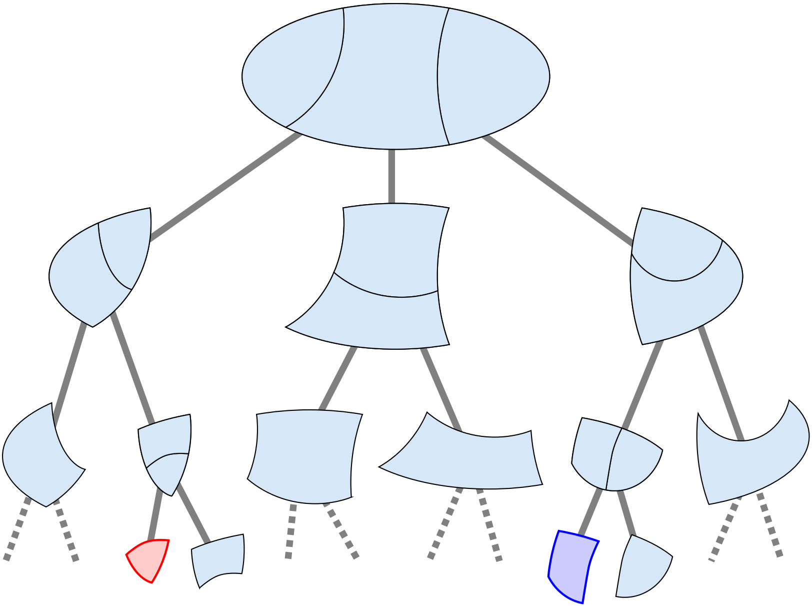
based on tree embedding

- **[R. 2008]**
  competitive ratio: $\mathcal{O}(\log n)$

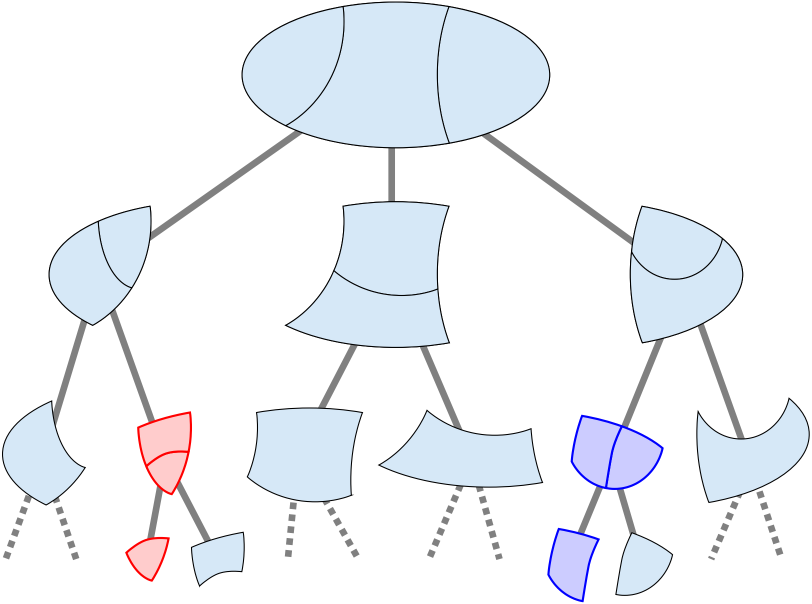
Underlying Path Selection Scheme



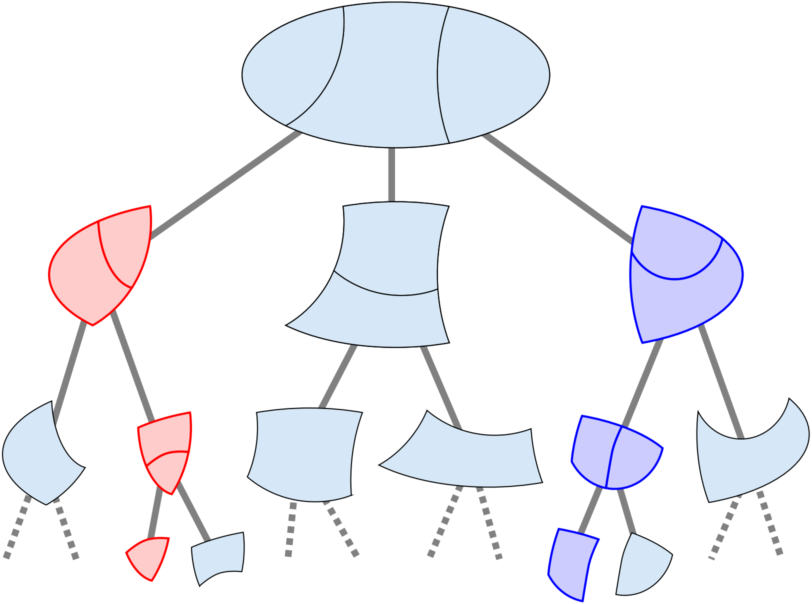
Underlying Path Selection Scheme



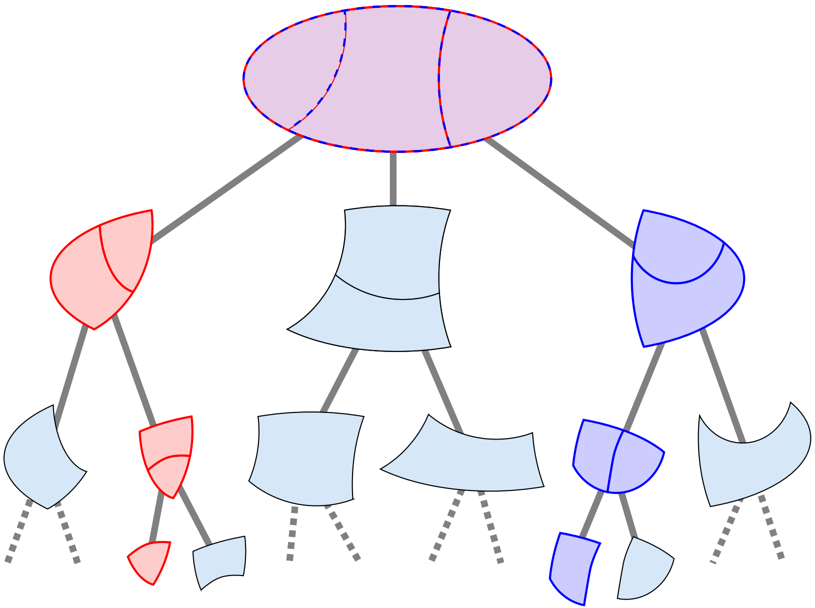
Underlying Path Selection Scheme



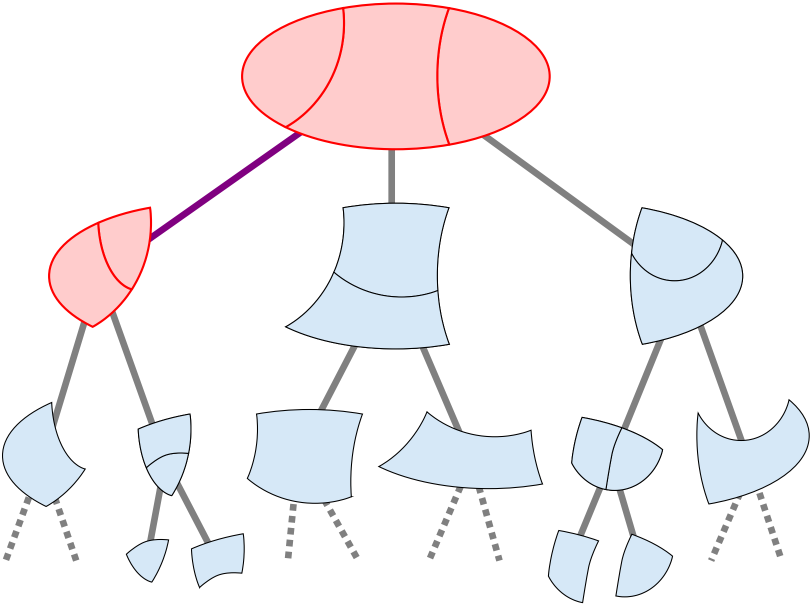
Underlying Path Selection Scheme



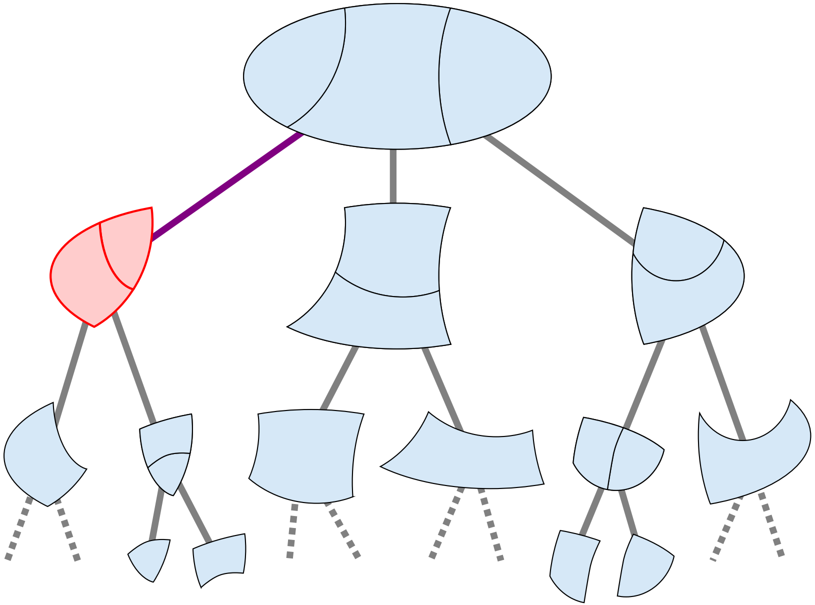
Underlying Path Selection Scheme



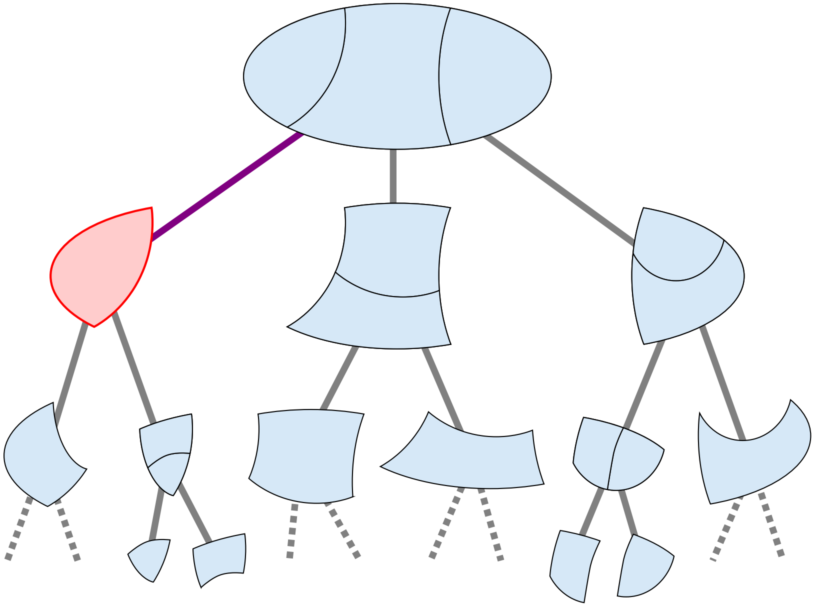
Underlying Path Selection Scheme



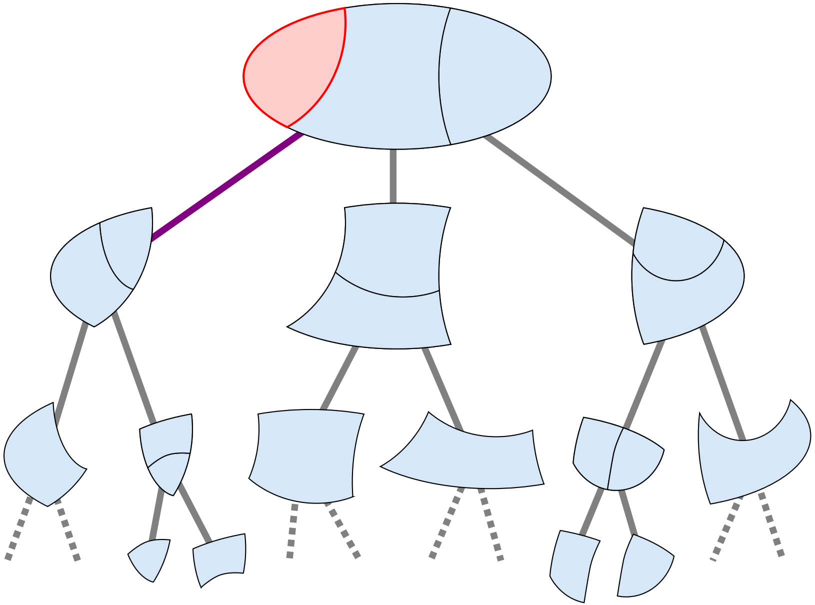
Underlying Path Selection Scheme



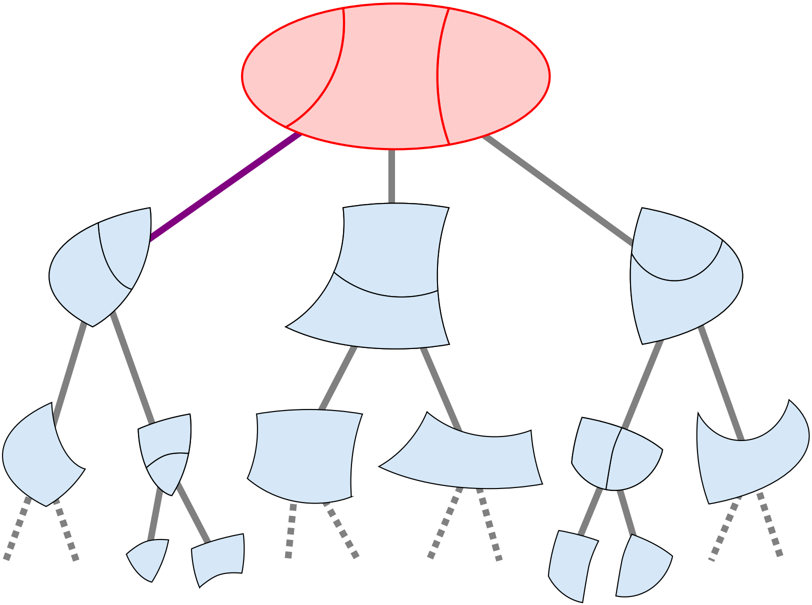
Underlying Path Selection Scheme



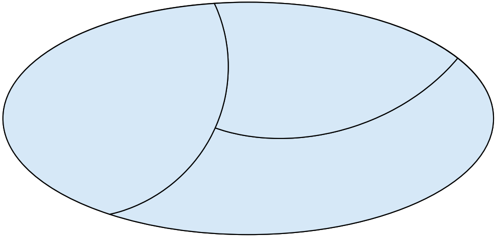
Underlying Path Selection Scheme



Underlying Path Selection Scheme



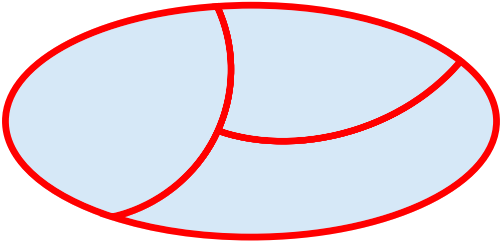
A Single Cluster S



Messages have following form:

1. route between cluster distribution and random border edge of sub-cluster, **or**
2. route between cluster distribution and random border edge of cluster

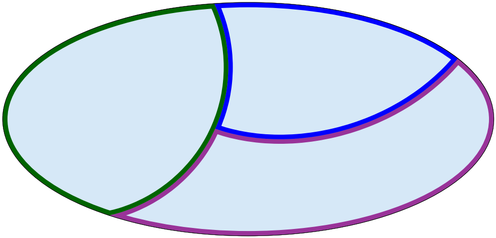
A Single Cluster S



Messages have following form:

1. route between cluster distribution and random border edge of sub-cluster, **or**
2. route between cluster distribution and random border edge of cluster

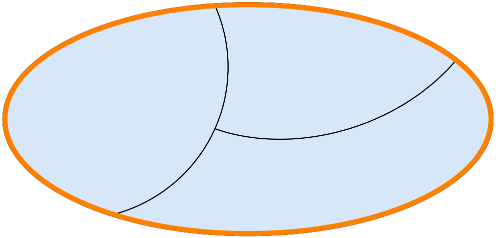
A Single Cluster S



Messages have following form:

1. route between cluster distribution and random border edge of sub-cluster, **or**
2. route between cluster distribution and random border edge of cluster

A Single Cluster S



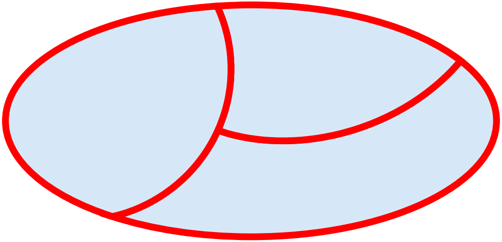
Messages have following form:

1. route between cluster distribution and random border edge of sub-cluster, **or**
2. route between cluster distribution and random border edge of cluster

A Single Cluster S

CMCF-problem for cluster

Every edge leaving a sub-cluster injects one unit of flow and sends it to a random of these edges.



Competitive Ratio of Hierarchical Approach

[R. 2002]
There is a hierarchical decomposition such that every CMCF-problem can be routed with congestion at most $\mathcal{O}(\log^2 n)$ inside the respective cluster.

$\Longrightarrow$ congestion $\mathcal{O}(\log^3 n)$ for all CMCF-problems

$\Longrightarrow$ competitive ratio $\mathcal{O}(\log^3 n)$

[Harrelson, Hildrum, Rao 2003]
There is a hierarchical decomposition such that all CMCF-problems together can be routed with congestion at most $\mathcal{O}(\log^2 n \log\log n)$.

$\Longrightarrow$ competitive ratio $\mathcal{O}(\log^2 n \log\log n)$

Variant A: naive encoding of CMCF-solutions

Encode a solution to the CMCF-problem for every cluster.

Every edge of the cluster-distribution gets two IDs, as it belongs to the border of two sub-clusters (or to one sub-cluster and the border of the whole cluster).

Every sub-cluster has a consecutive range of IDs.

A vertex of a cluster stores the ID-ranges of all sub-clusters and the ID-range for the whole cluster (border). Note that ID-ranges for different CMCF-problems are different.

Variant A: naive encoding of CMCF-solutions

Label node by its path from the root in the decomposition tree.

Routing:

- given source and destination label compute path in the decomposition tree
- send packet to random edge incident to source s (i.e., distribute according to cluster distribution of $\{s\}$)
- for every upward edge ((S_i, S_{i+1}))
  - cluster distribution of $S_i \longrightarrow$ border distribution of S_i routed according to CMCF-solution for S_i
  - border distribution of $S_i \longrightarrow$ cluster distribution of S_{i+1} routed according to CMCF-solution for S_{i+1}
- for every downward edge ((S_{i+1}, S_i))
  - cluster distribution of $S_{i+1} \longrightarrow$ border distribution of S_i routed according to CMCF-solution for S_{i+1}
  - border distribution of $S_i \longrightarrow$ cluster distribution of S_i routed according to CMCF-solution for S_i

Variant A: naive encoding of CMCF-solutions

Encode an optimum all-to-all multicommodity flow for every cluster with precision ϵ.

- competitive ratio: $\mathcal{O}(\text{height}(T) \log^2 n)$
- labels encode path in the decomposition tree;
  label size: $\mathcal{O}(\text{height}(T) \log(\deg(T)))$
- header encodes path between source and target in the tree; id of target in current routing step
  header size: $\mathcal{O}(\text{height}(T) \log(\deg(T))) + \mathcal{O}(\log m)$
- a node stores for every flow a probability distribution over outgoing edges;
  table size: $\mathcal{O}(m^2 \,\text{height}(T) \deg(v) \log(1/\epsilon))$ **very poor**

  in addition it stores the ranges for sub-clusters;
  size: $\deg(T)\,\text{height}(T) \log(m)$

Variant B: encode single-commodity flows to sub-clusters

Encode a flow from every sub-cluster border to the cluster-distribution (and back) ($\leq 2\deg(T)$ flows for every cluster)

- competitive ratio: $\mathcal{O}(\text{height}(T)\deg(T)\log^2 n)$
- labels encode path in the decomposition tree;
  label size: $\mathcal{O}(\text{height}(T)\log(\deg(T)))$
- header encodes path between source and target in the tree;
  header size: $\mathcal{O}(\text{height}(T)\log(\deg(T)))$
- a node stores for every flow a probability distribution over outgoing edges;
  table size: $\mathcal{O}(\deg(T)\,\text{height}(T)\deg(v)\log(1/\epsilon))$

Variant C: hypercube embedding (unweighted graph)

Edges are assigned IDs as in Variant A. Assume that the number of IDs is 2^d.

Embed d-dimensional hypercube by solving a CMCF-problem. Can be embedded with congestion $\mathcal{O}(d \log^2 n)$ as any permutation can be embedded with congestion $\mathcal{O}(\log^2 n)$.

Apply randomized rounding:

- decompose the flow for every commodity into a distribution over paths
- pick a single path from this distribution
- with high probability the load on any edge increases by at most an additive $\mathcal{O}(\log n)$

Variant C: hypercube embedding

When routing from ID 1 to ID 2 we route along edges of the hypercube using a random intermediate destination.

This induces constant expected load on an edge of the hypercube (provided the overall demand can be routed with congestion 1).

Variant C: hypercube embedding

$d = \mathcal{O}(\log m)$ is dimension of cube

- competitive ratio: $\mathcal{O}(\text{height}(T) d \log^2 n)$
- labels encode path in the decomposition tree;
  label size: $\mathcal{O}(\text{height}(T) \log(\deg(T)))$
- header encodes path between source and target in the tree;
  in addition it stores intermediate target(s) in the cube
  header size: $\mathcal{O}(\text{height}(T) \log(\deg(T))) + \mathcal{O}(d)$
- a node stores its hypercube IDs and for every path a path-id and an outgoing edge;
  table size:
  $\mathcal{O}(\text{height}(T) \deg(v) (\log(m) + d \log^2 n \cdot \log(\deg(v))))$

  in addition it stores the ranges for sub-clusters...

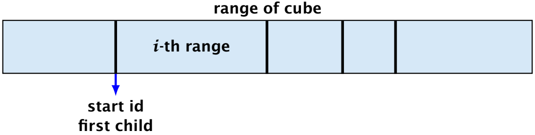
Variant C: hypercube embedding

Round every range to a power of 2.

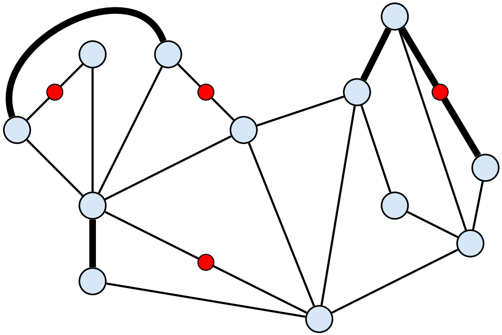
Sort the subclusters according to the size of their range.

Let $R \leq \deg(T) + 1$ denote the number of ranges; (number of sub-clusters + 2). Let $S \leq \log m$ denote the number of different range classes.

We store separators between range classes, and for every separator the start ID of the range class. Requires at most $\mathcal{O}(\log m \cdot (\log(\deg(T)) + \log m))$ bits.



Variant D: weights 1 and W



Task: embed all-to-all flow between subset of edges

Challenge:

- Randomized rounding of all-to-all flow generates path of weight 1.
- A heavy edge may see W of these paths.

Variant D: weights 1 and W

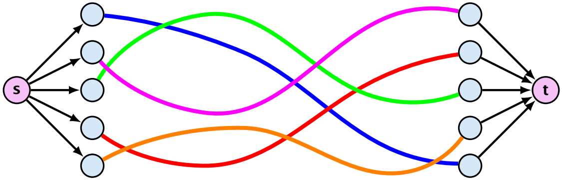
Sub-problem

Route all-to-all between subset of light vertices.

Idea

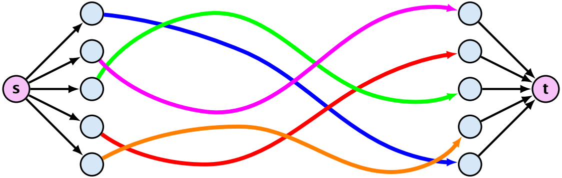
Use cut-matching game [KRV] to embed expander between light vertices.

Cut-matching game embeds an expander as a set of $\mathrm{polylog}(n)$ arbitrary matchings between subsets.

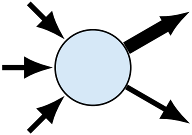


Variant D: weights 1 and W

We can store one-directional routing paths along a matching very efficiently.

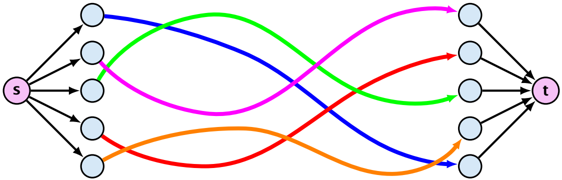


The paths arise from decomposing an integral single-commodity flow.

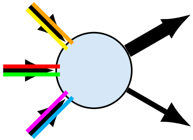


Variant D: weights 1 and W

We can store one-directional routing paths along a matching very efficiently.

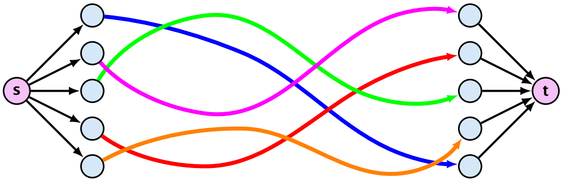


The paths arise from decomposing an integral single-commodity flow.

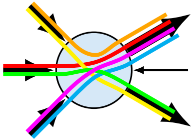


Variant D: weights 1 and W

We can store one-directional routing paths along a matching very efficiently.



The paths arise from decomposing an integral single-commodity flow.



Variant D: weights 1 and W

Instead of embedding a matching we embed two directional matchings in every round.

This is sufficient to get an expander. Congestion $\mathcal{O}(\log^2 n)$ for every round of the cut-matching game.

In this expander we embed a hypercube; this can be done with congestion $\mathcal{O}(\log^2 n)$ and path of logarithmic length (inside the expander).

We can store this embedding by storing the whole path at every source. In total $\mathcal{O}(\log m)$ bits for every incident hypercube edge.

Variant D: weights 1 and W

If the light vertices are in the majority the heavy vertices can first route to light vertices (via a multi-commodity flow) and use the hypercube there.

Can also be extended to the case when the heavy vertices are in the majority.

Open Problems:

- General weighted graphs?
- For which graphs do we have decomposition trees with small degree?
- Generally, what is the loss in quality if we want to establish a multi-commodity flow solution with small routing tables?
- Name-independent case?