
Optimal Oblivious Path Selection on the Mesh

Costas Busch Malik Magdon-Ismael **Jing Xi**

Rensselaer Polytechnic Institute



Outline

- Introduction
- Related Work and Our Contributions
- Results on Mesh Networks
- Future Work

Outline

- Introduction
- Related Work and Our Contributions
- Results on Mesh Networks
- Future Work

Oblivious Routing Algorithms

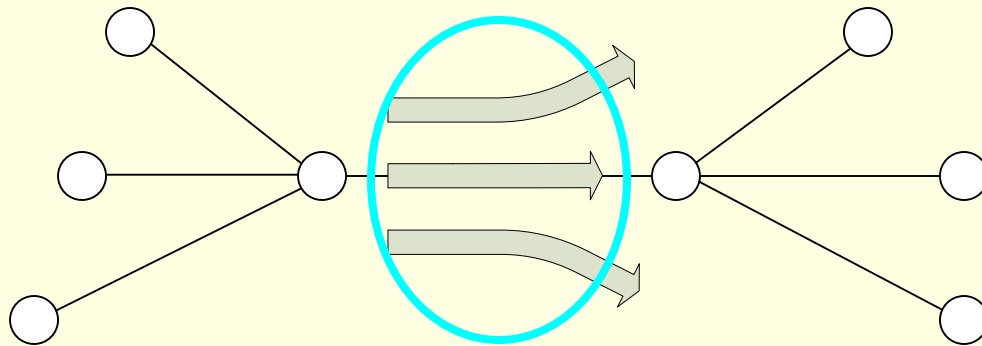
- Routing algorithms provide paths for packets in the network.
- A routing algorithm is **oblivious** if the path of every packet is specified independently of the paths of the other packets.

Benefits of Oblivious Routing

- Distributed algorithm
- Capable of solving online routing problems, where packets continuously arrive in the network.

Congestion C

- Congestion C: the maximum number of paths that use any edge in the network

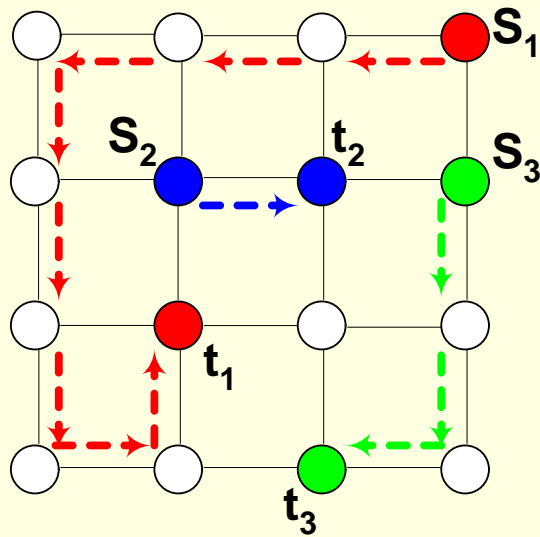


The total number of paths that use this edge

Congestion C takes the maximum over all edges.

Dilation D

- Dilation D: the maximum length of any path

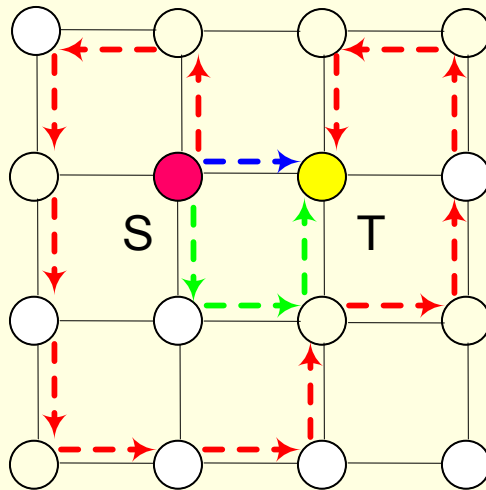


The length of the red path is 8 hops

Dilation D takes the maximum over all paths.

Stretch

- $dist(s,t)$: the length of the shortest path from s to t .
- $stretch(p_i) = |p_i|/dist(s_i, t_i)$.
- $stretch(P) = \max_i stretch(p_i)$.



$$dist(s,t) = 1$$

$$stretch(p_i) = 1$$

$$stretch(p_i) = 3$$

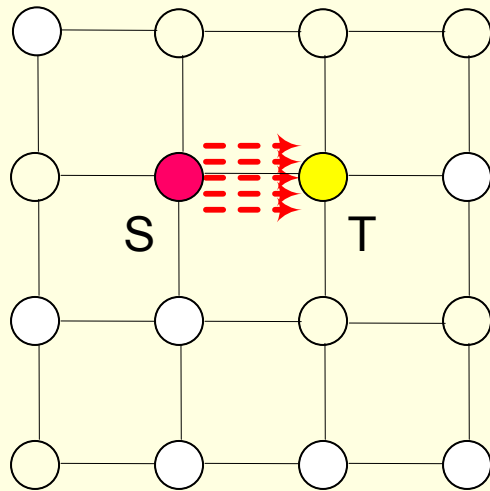
$$stretch(p_i) = 13$$

$$stretch(P) = 13$$

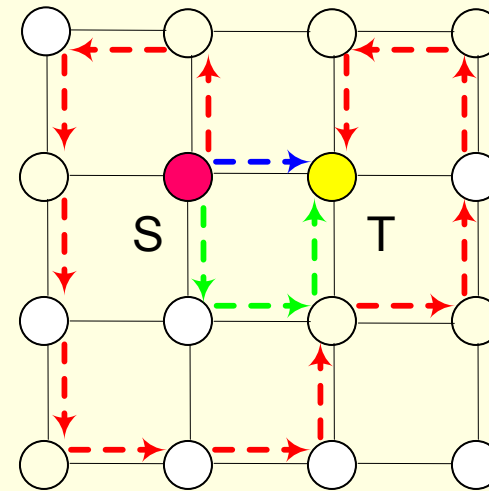
Motivations

- A trivial lower bound on transfer time: $\Omega(C+D)$
- So far, the oblivious algorithms focused on minimizing the congestion while ignoring the dilation
- An open question is whether C and D can be controlled simultaneously.

Motivations



Good Stretch but Bad Congestion!



Good Congestion but Bad Stretch!

We want to find such set of paths that minimize $C+D$!

Related Work

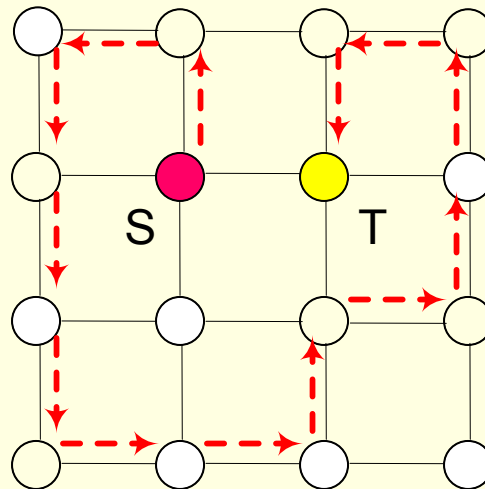
- Introduction
- **Related Work and our contributions**
- Results on Mesh Networks
- Future Work

Related Work

- Maggs, et al. in [FOCS1997] presented an oblivious algorithm with optimal congestion $O(C^* \log n)$ on Mesh.

C^* : Optimal Congestion for any routing algorithm

- Problems: The stretch factor is unbounded.



Related Work

- Y.Azar et al. in *[STOC 2003]*
Marcin Bienkowski et al. in *[SPAA 2003]*
Chris Harrelson et al. in *[SPAA 2003]*
Harald Racke et al. in *[FOCS 2002]* gave progressively better oblivious algorithms with near optimal congestion on general networks
- Problems: Unbounded stretch
- Our algorithm: Constant stretch without sacrificing on the congestion

Related Work

- A.Srinivasan et al. in [STOC 1997] presented near optimal algorithm but the algorithm is *offline and non-oblivious*
- Problems: Require knowledge of the traffic distribution and generally do not scale well

Our Contributions

- For 2-dimensional Mesh, our path selection algorithm has congestion $O(C^* \log n)$, and stretch $\Theta(1)$.
- For d -dimensional Mesh, our path selection algorithm has congestion $O(dC^* \log n)$, and stretch $O(d^2)$.
- Number of random bits required per packet is within $O(d)$ of the optimal.

Our Contributions

- First oblivious algorithm to control C+D simultaneously!

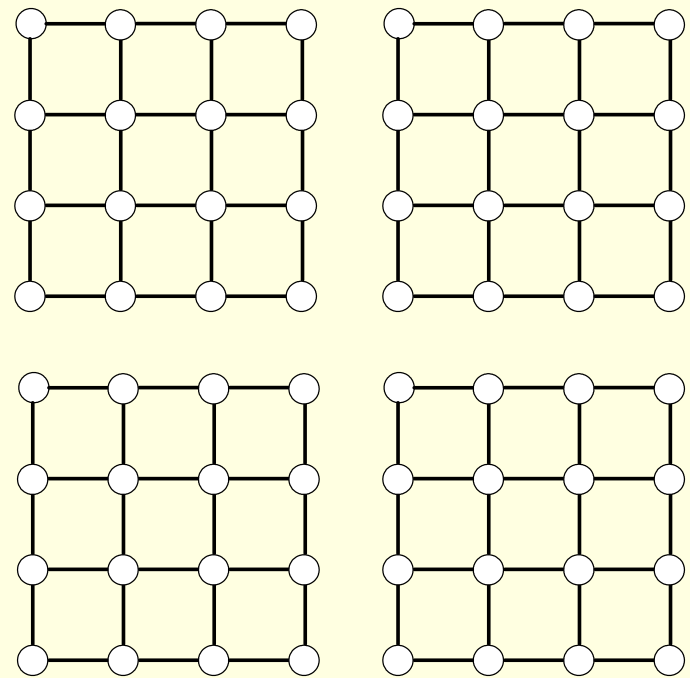
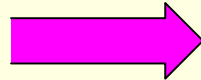
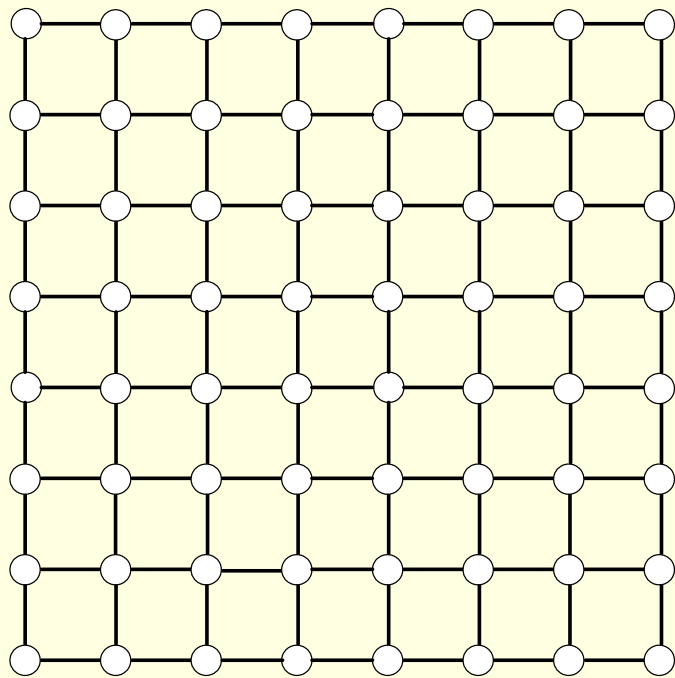
	Congestion	Stretch
Maggs' algorithm	$O(dC \cdot \log n)$	$O(n)$
Our algorithm	$O(dC \cdot \log n)$	$O(d^2)$

- Randomization: Number of random bits required per packet is within $O(d)$ of the optimal.

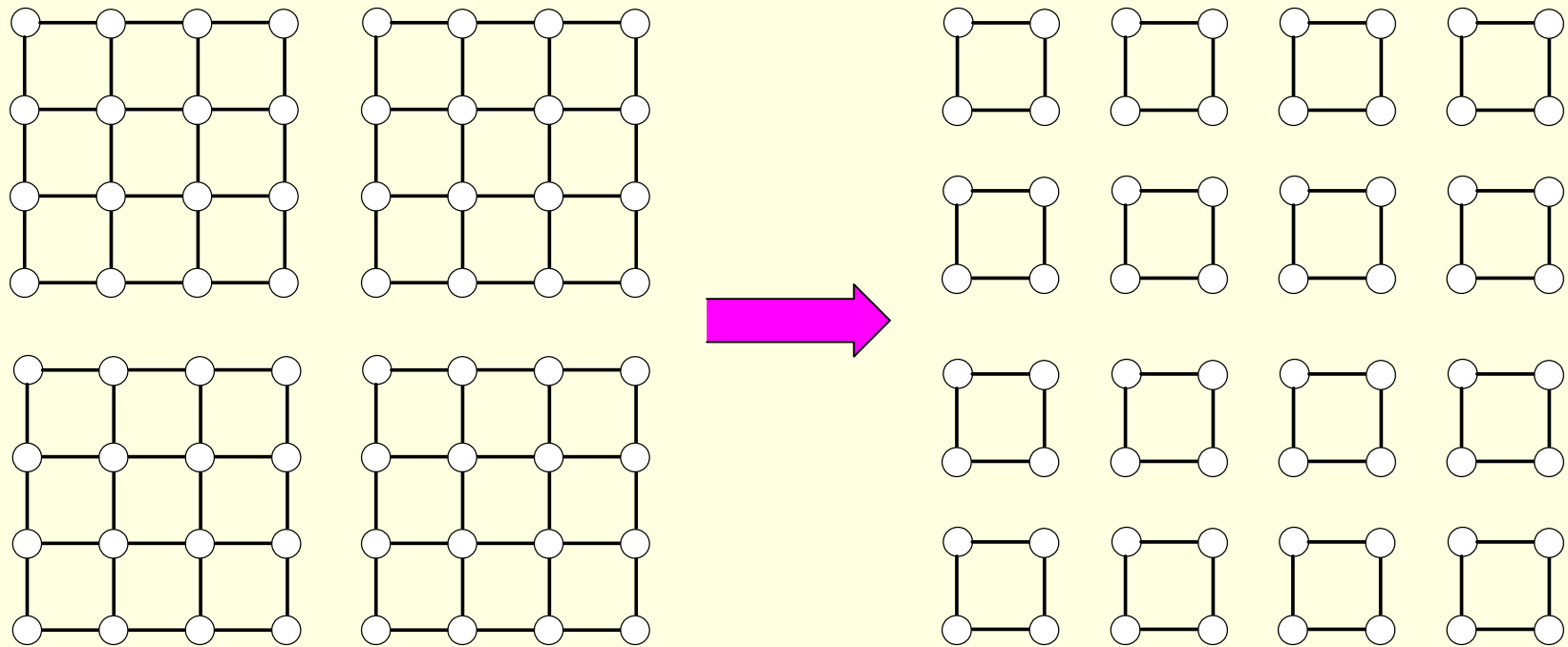
Oblivious Routing Algorithm on Mesh

- Introduction
- Related Work and Our Contributions
- **Results on Mesh Networks**
- Future Work

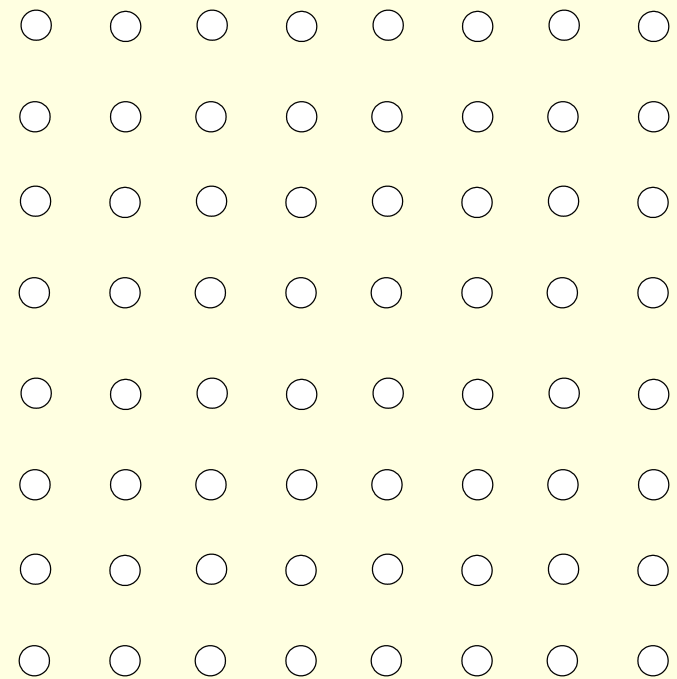
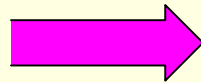
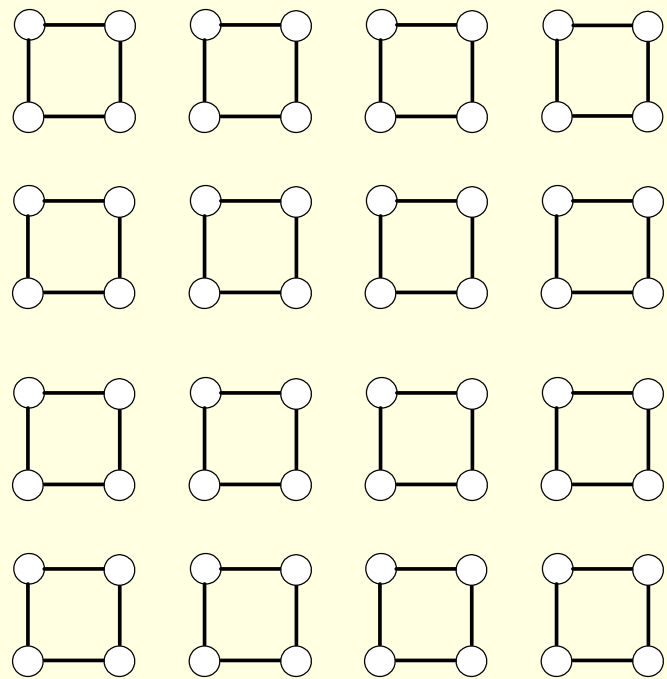
Type I Mesh Decomposition



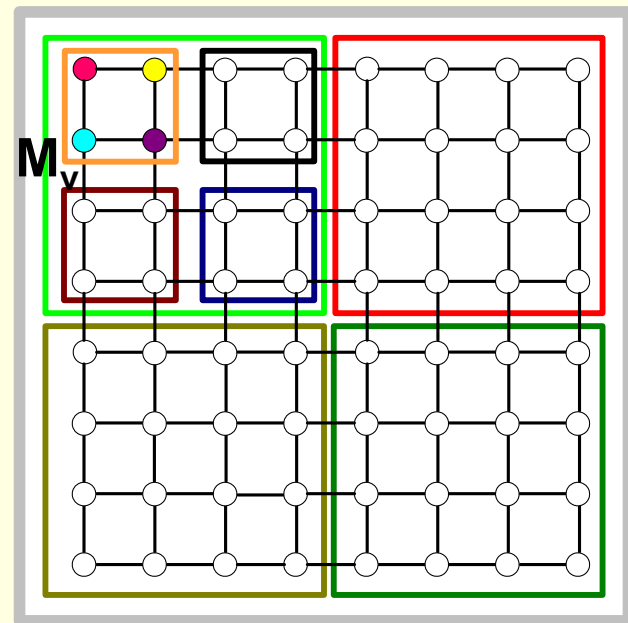
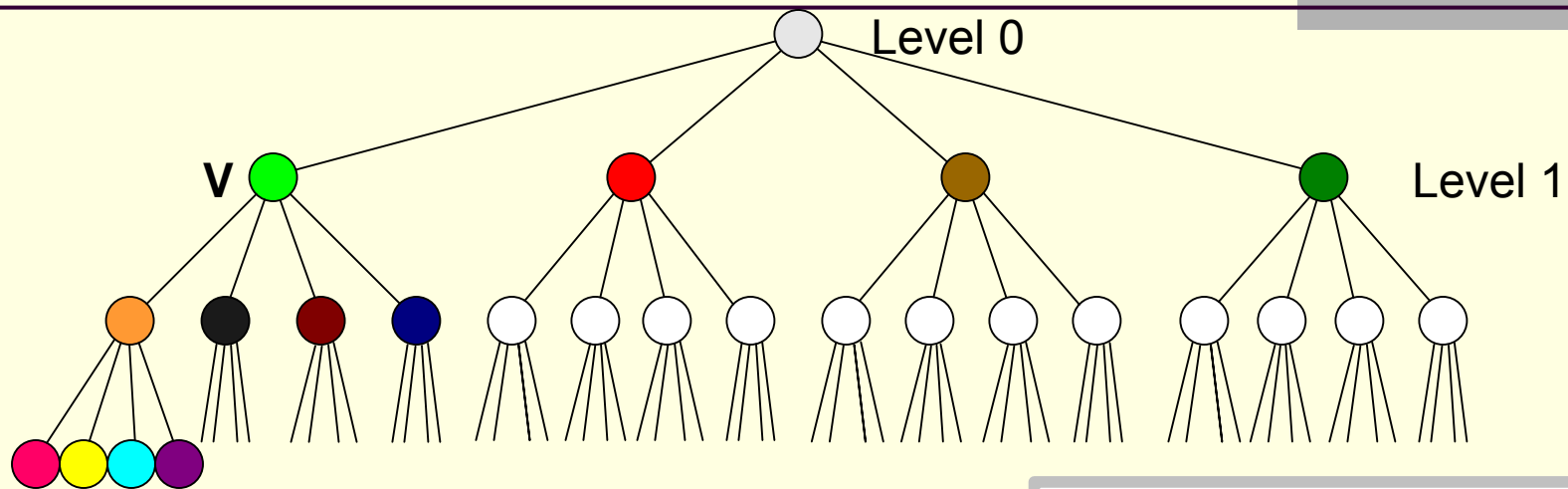
Type I Mesh Decomposition



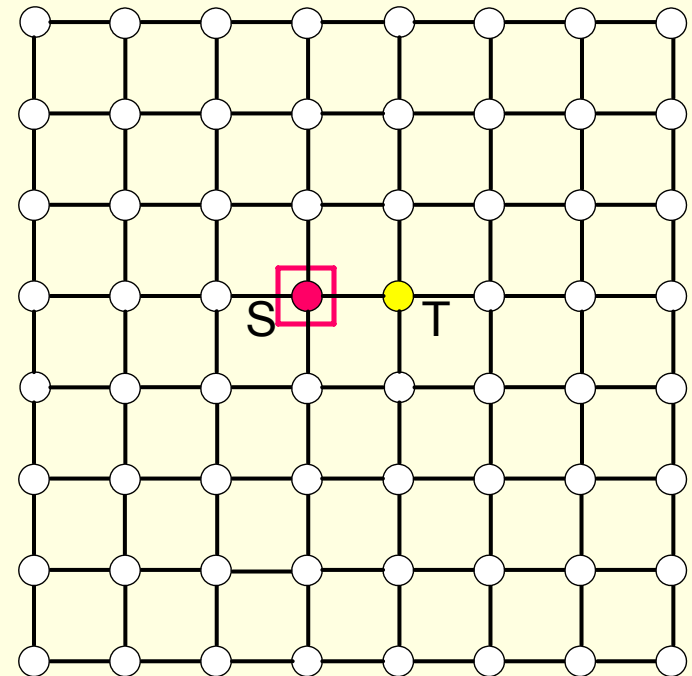
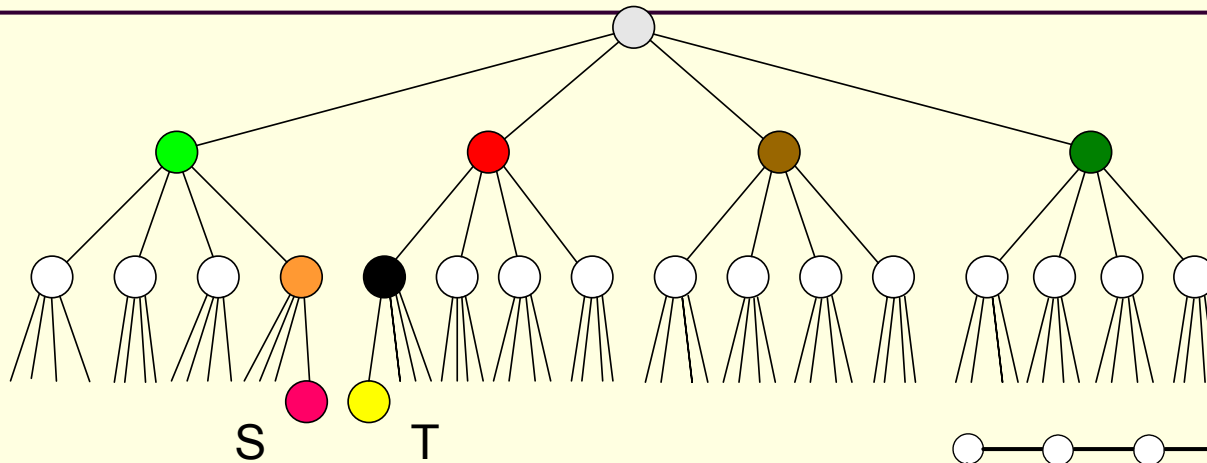
Type I Mesh Decomposition



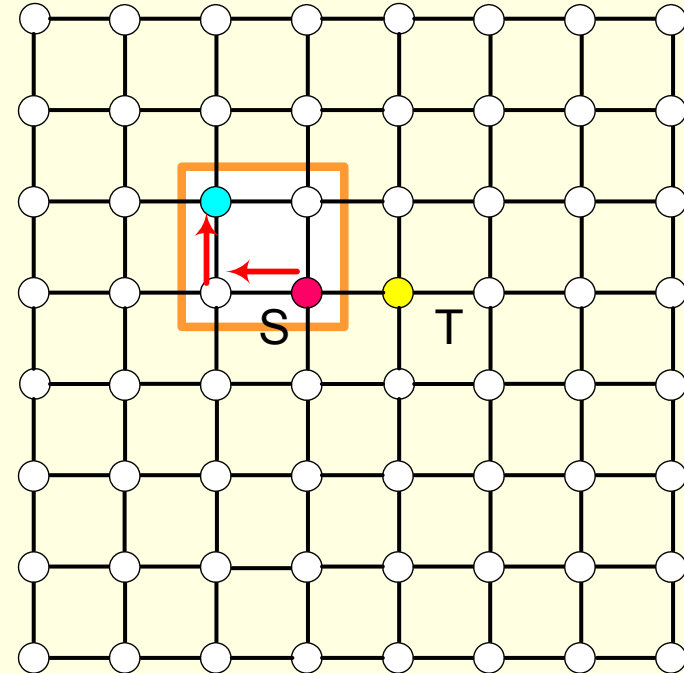
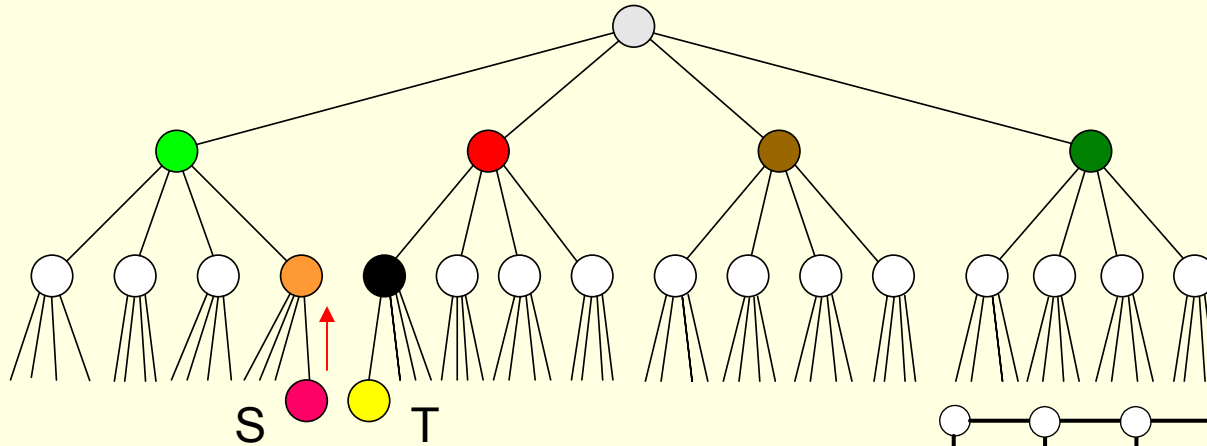
Access Graph for Type I Mesh



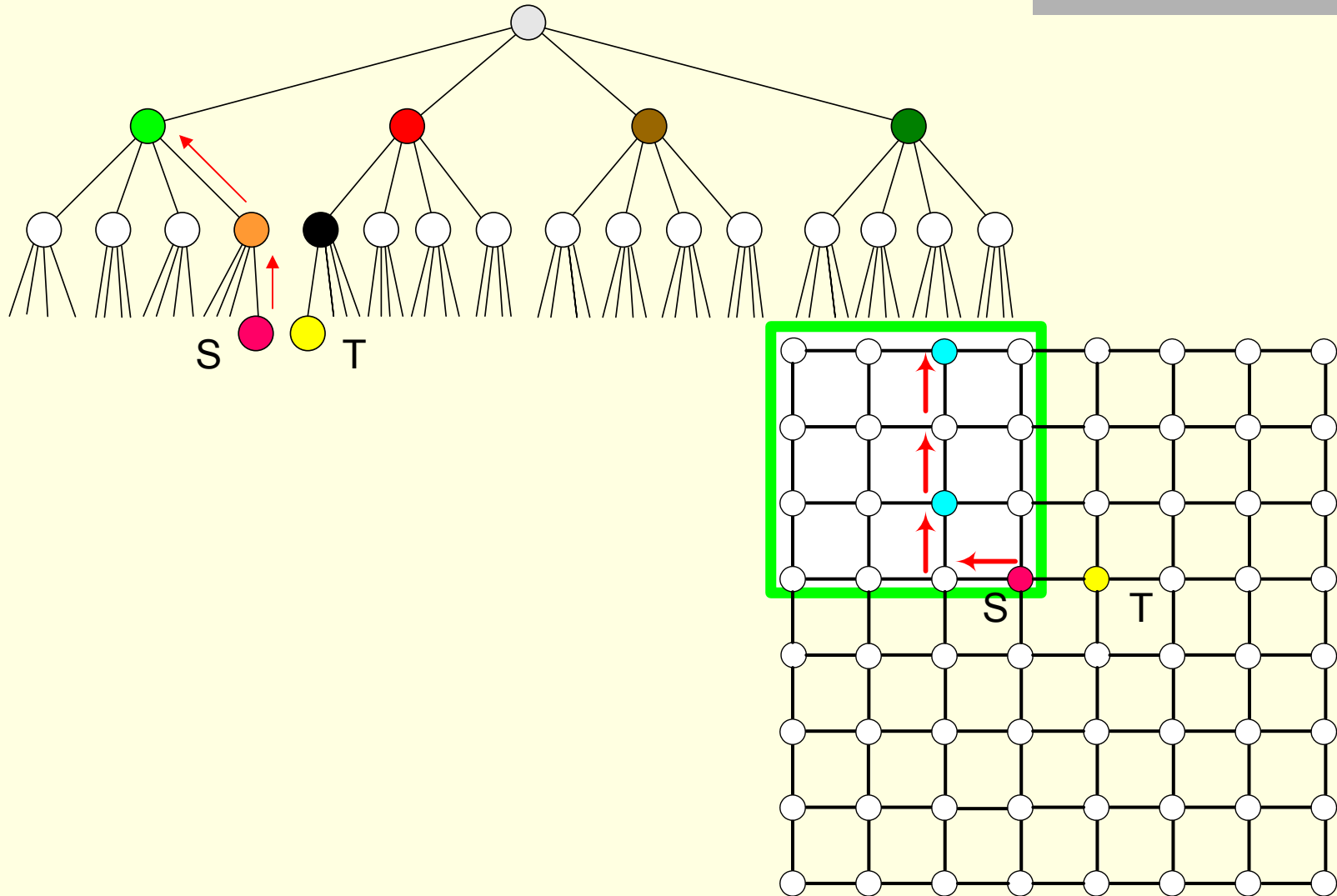
Routing for Type I Mesh Decomposition



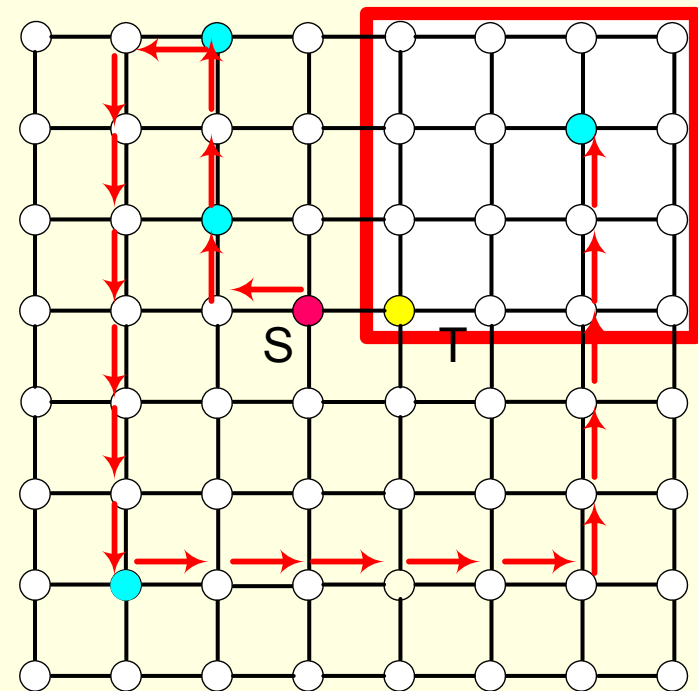
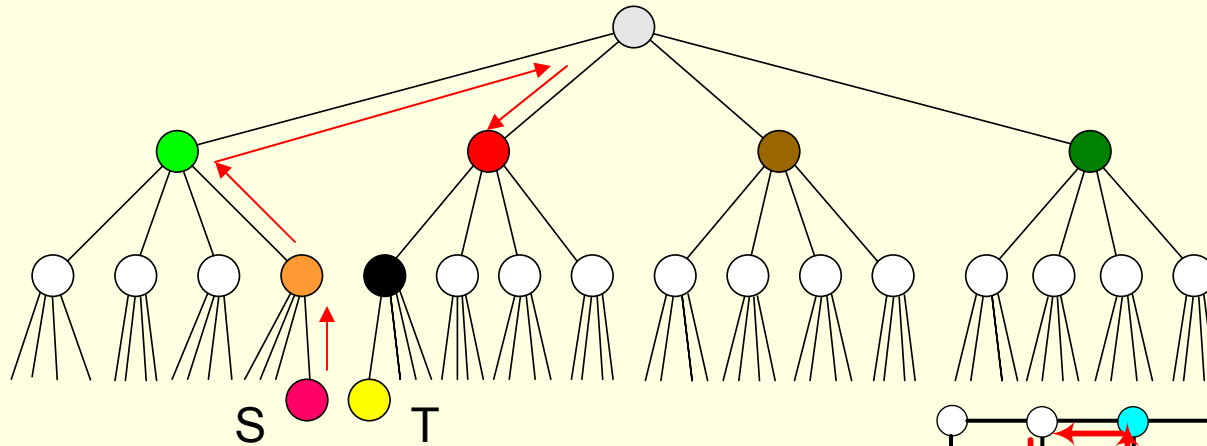
Routing for Type I Mesh Partition



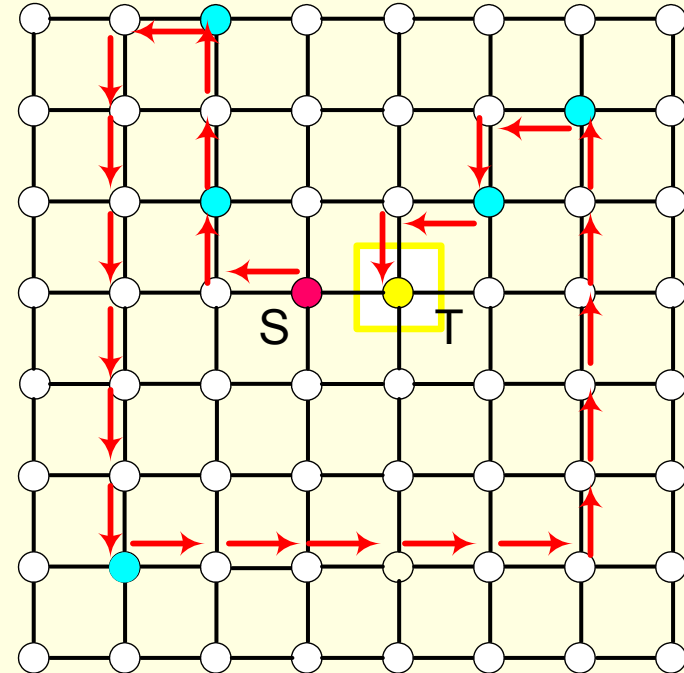
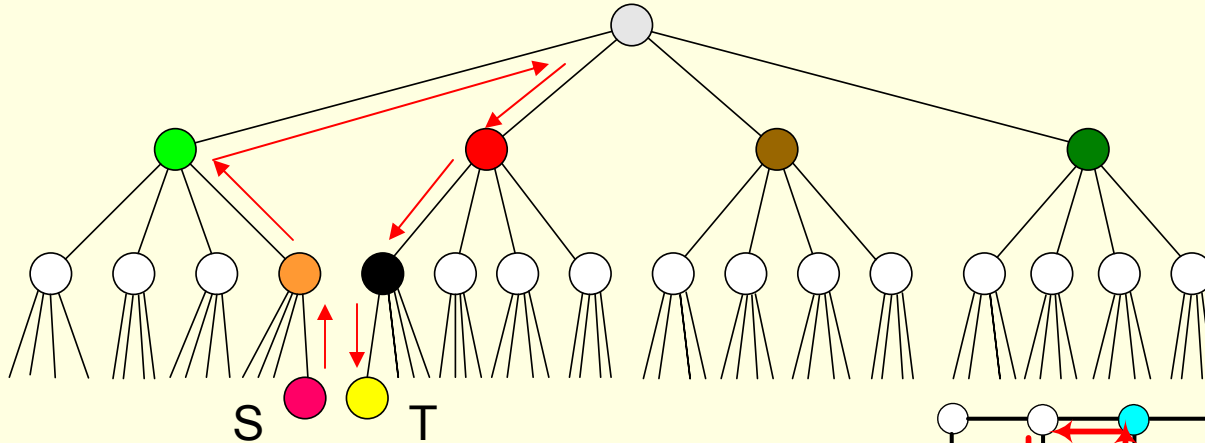
Routing for Type I Mesh Partition



Routing for Type I Mesh Partition

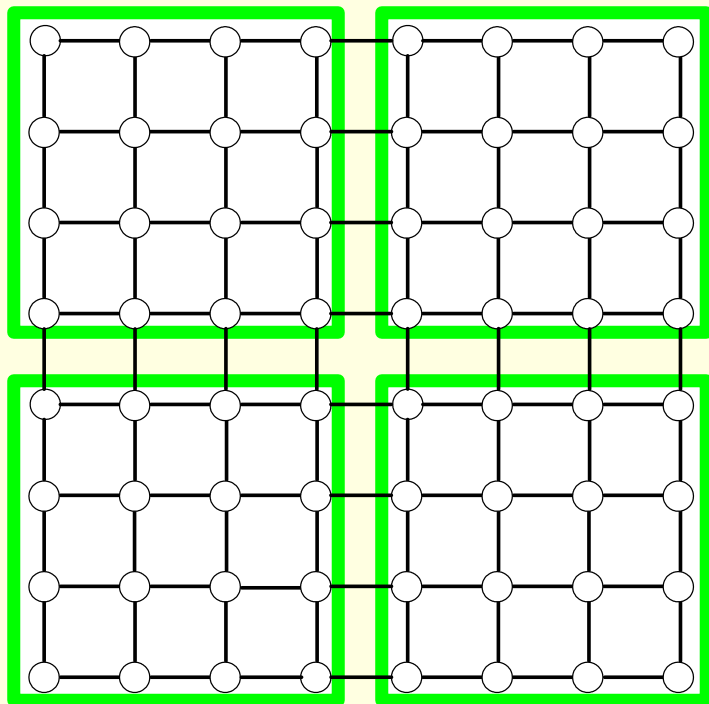


Routing for Type I Mesh Partition

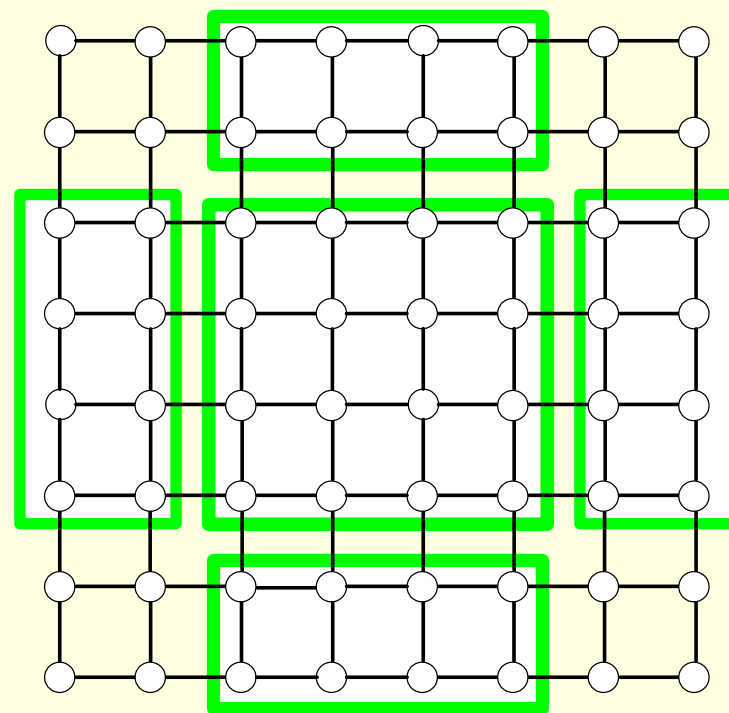


Even for neighbor nodes, it may take a path with length $O(n)$!
Stretch = $O(n)$!
 n : # of nodes in the network

Type II Mesh Decomposition

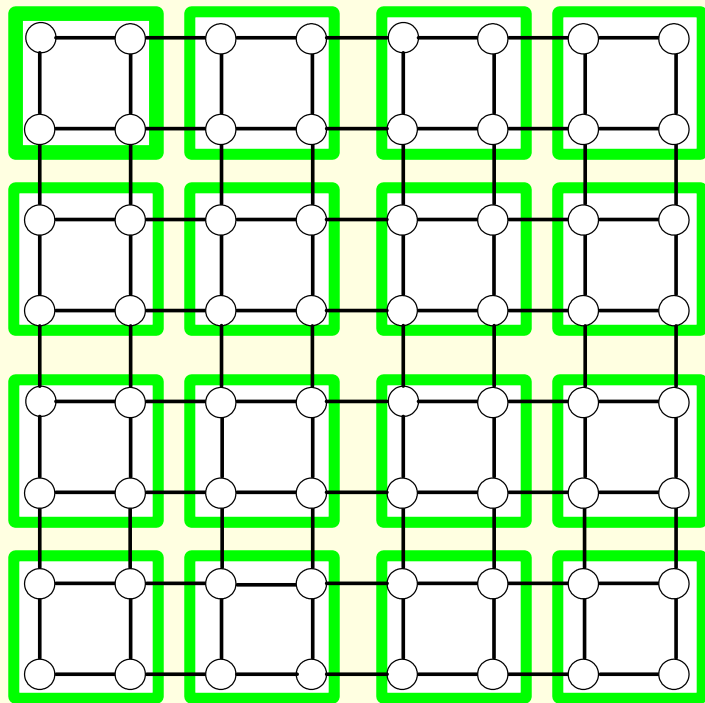


Level 1, Type 1

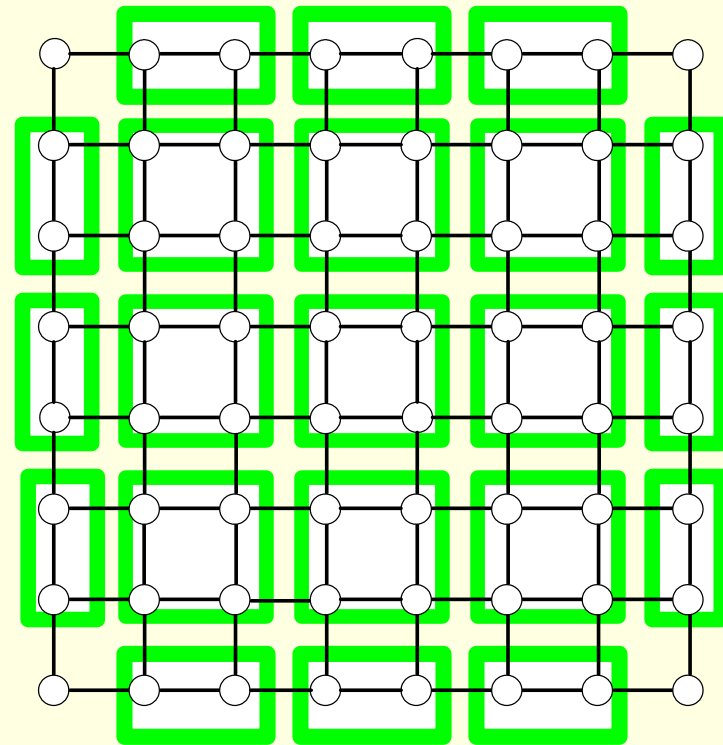


Level 1, Type 2

Type II Mesh Decomposition

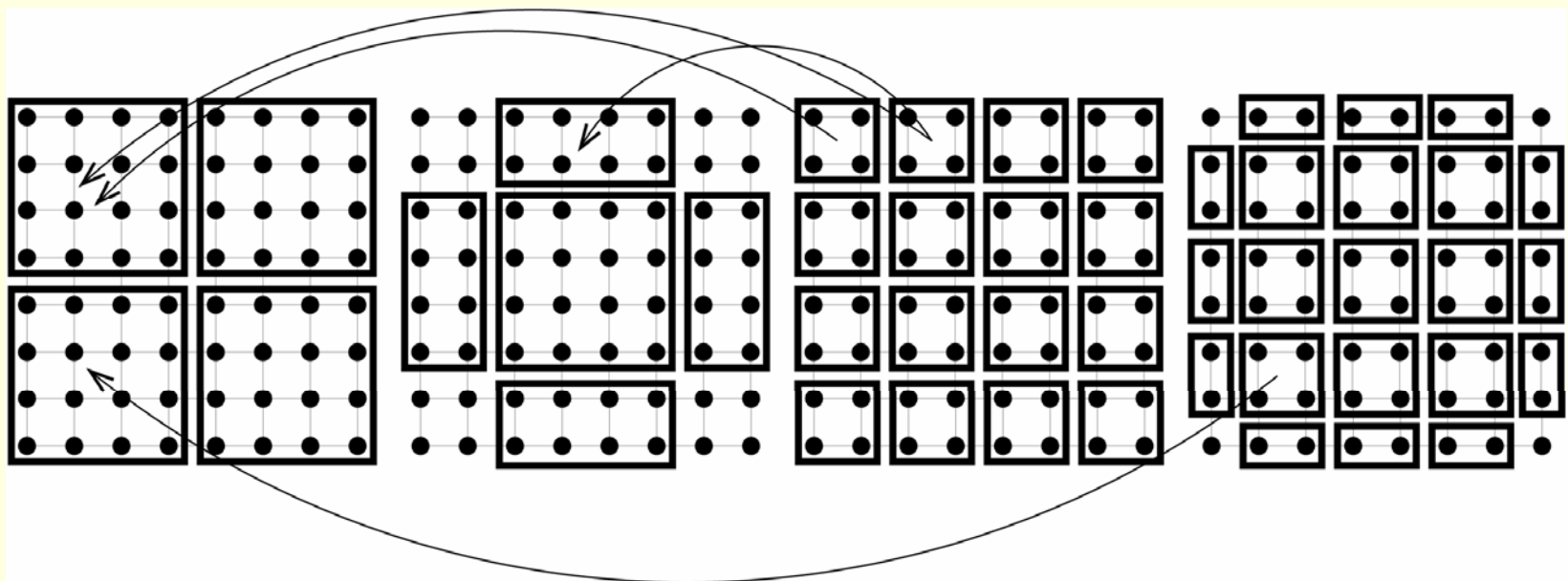


Level 2, Type 1



Level 2, Type 2

Type II Mesh Decomposition



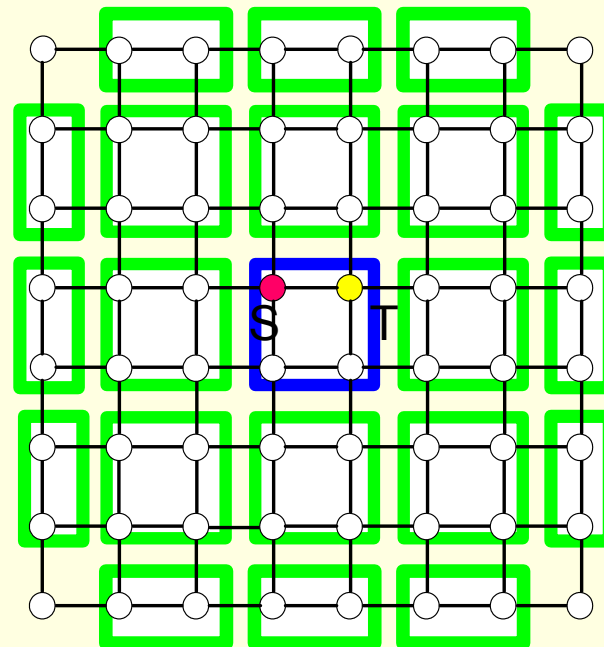
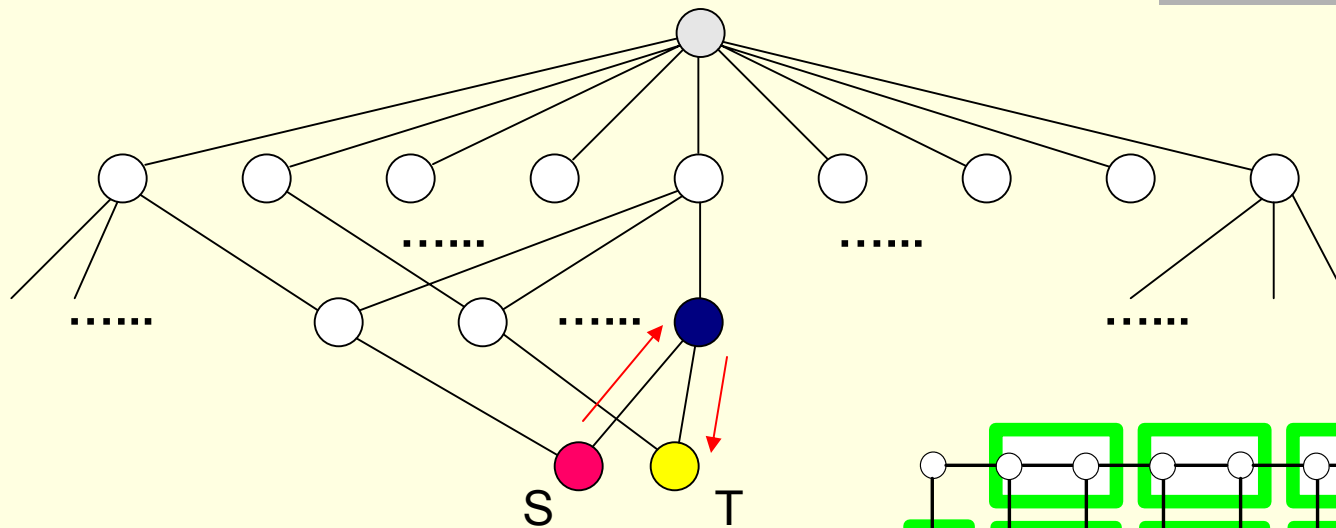
■ Level 1 type 1.

Level 1, type 2.

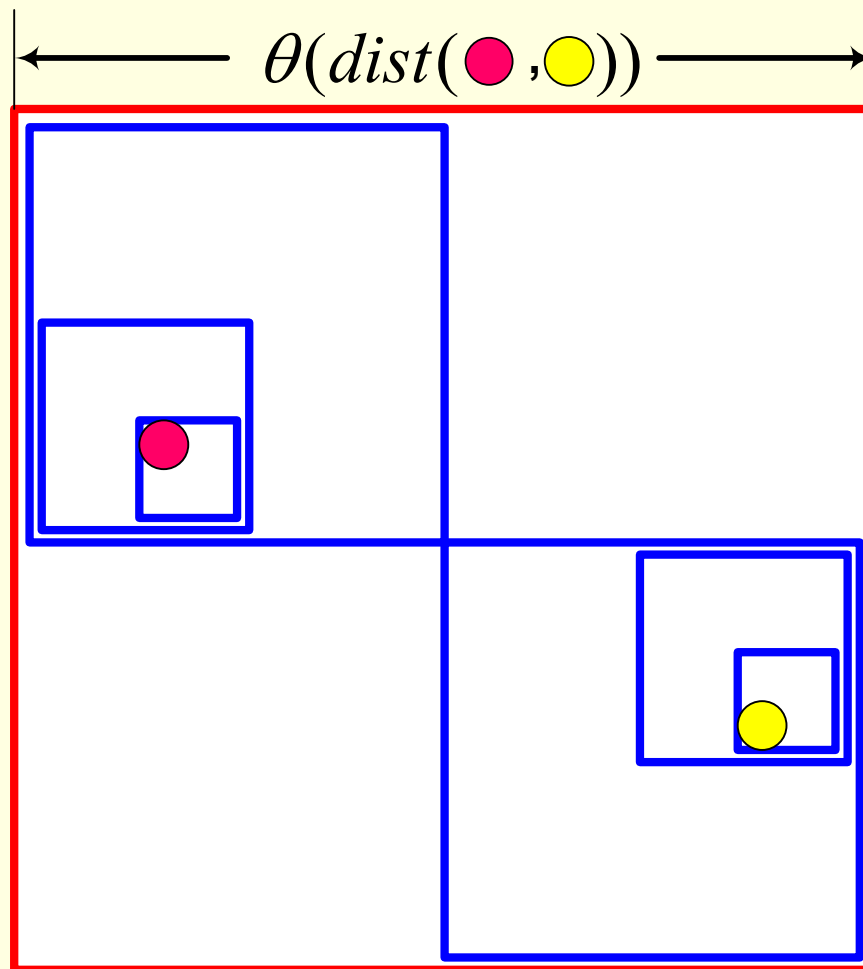
Level 2, type 1.

Level 2, type 2.

An Routing Example



Analysis on Stretch

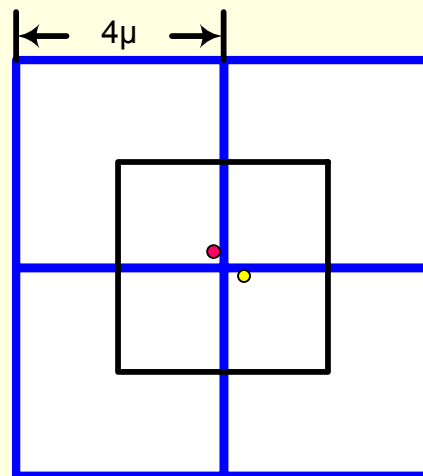
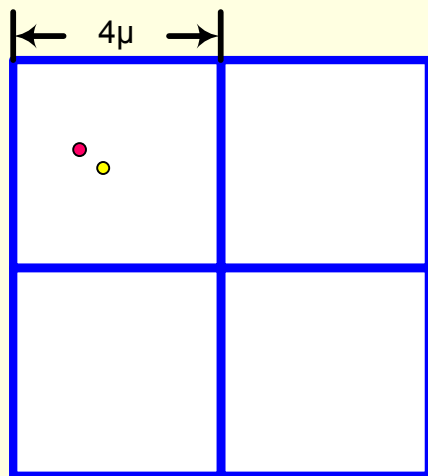


Bridge Submesh:
Type I or Type II.

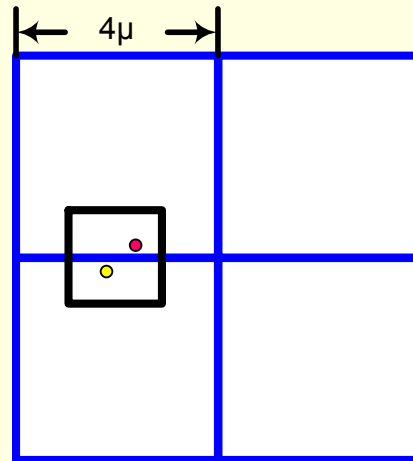
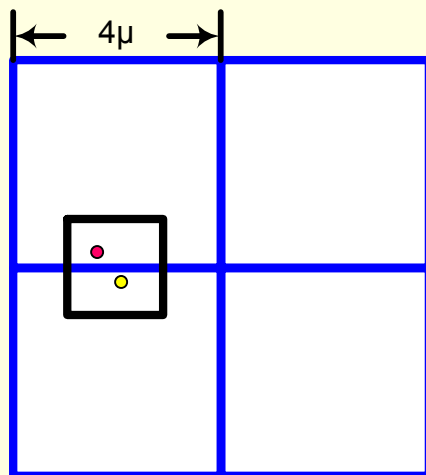
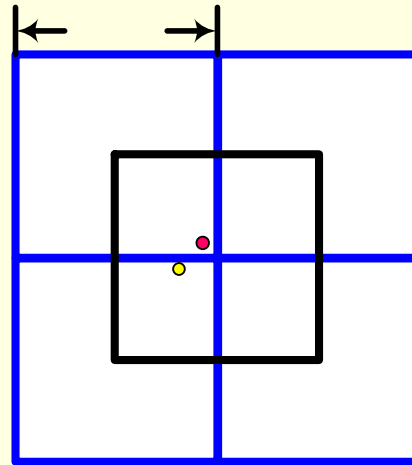
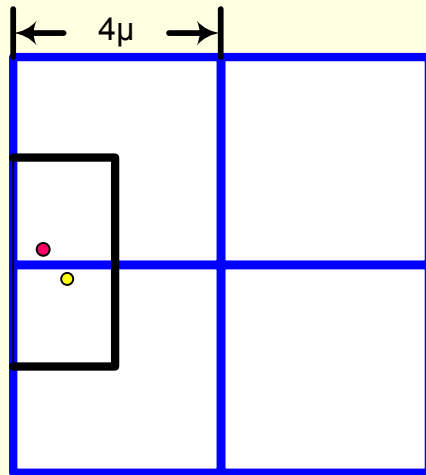
Only 1 Type II
submesh is ever
needed

Analysis on Stretch

$$\mu = 2^{\lceil \log \text{dist}(s,t) \rceil} \geq \text{dist}(s,t)$$



Analysis on Stretch



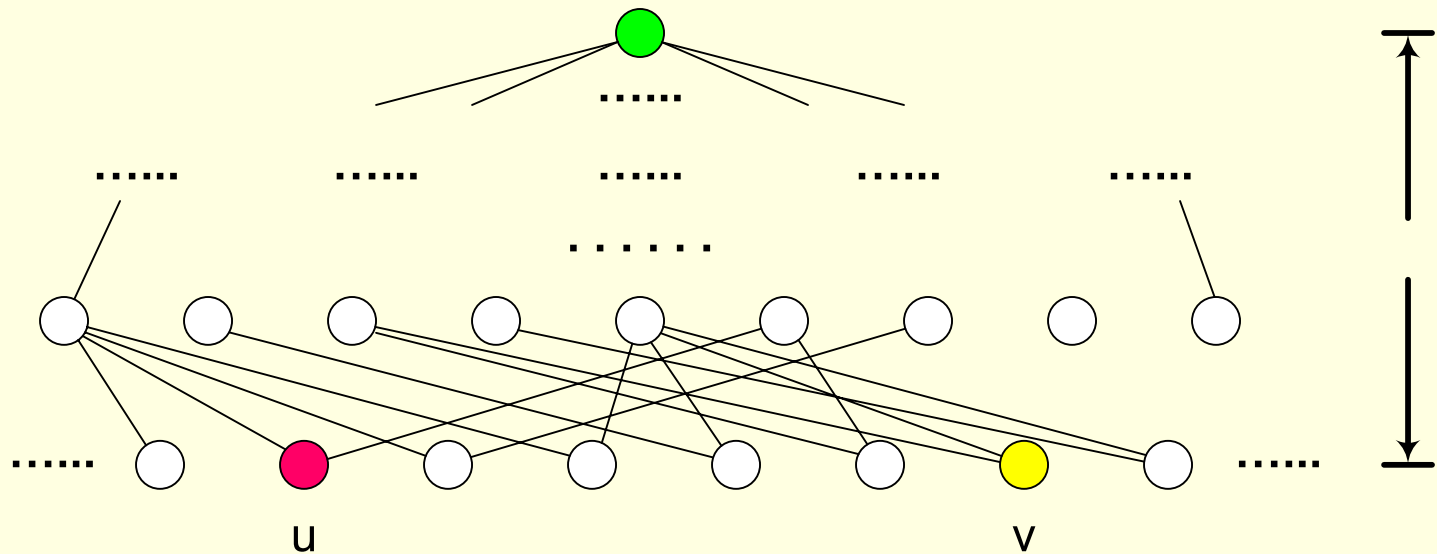
In all cases, \bullet , \bullet are contained in a submesh of side length 4μ , and each follows type I submeshes only until reach the bridge.

■ **Theorem** For any two distinct nodes s and t of the mesh, $stretch(p(s, t)) \leq 64$.

4μ

Analysis on Stretch

- **Lemma** The deepest common ancestor of two leaves u and v has height at most $\lceil \log \text{dist}(g(u), g(v)) \rceil + 2$.



Analysis on Edge Congestion

- Subpath uses edge e with probability at most $2/m_1$
- P' : set of paths from M_1 to M_2 or vice-versa
 $C'(e)$: congestion caused by P'

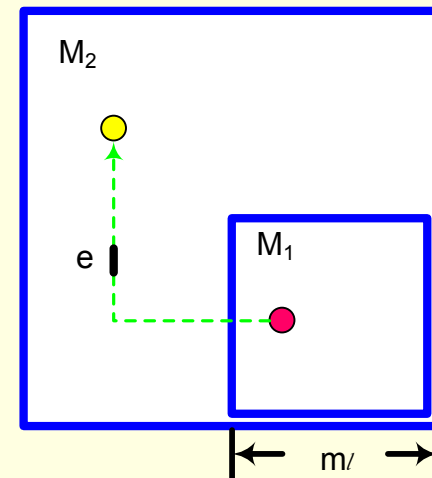
- $E[C'(e)] \leq 2 |P'| / m_1$

$$B \geq |P'| / out(M_1)$$

$$out(M_1) \leq 4m_1$$

$$C^* \geq B$$

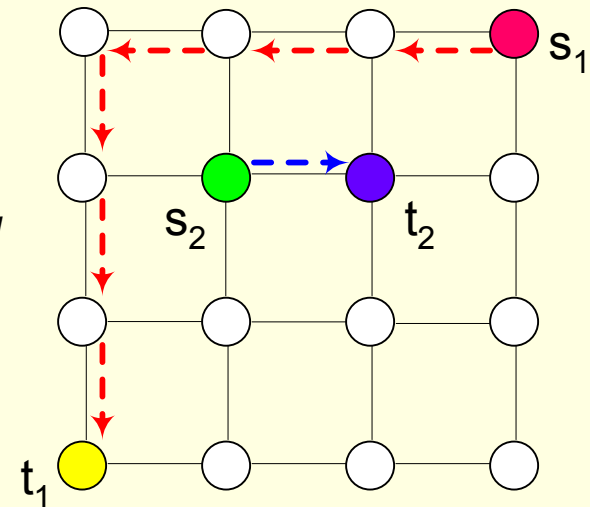
$$\Rightarrow E[C'(e)] \leq 8C^*.$$



Analysis on Edge Congestion

- **Theorem** The deepest common ancestor of two leaves u and v has height at most $\lceil \log \text{dist}(g(u), g(v)) \rceil + 2 < \log D^* + 3$.
- $D^* = \max_i \text{dist}(s_i, t_i)$

$$\text{dist}(s_2, t_2) = 1$$
$$D^* = 6$$



Analysis on Edge Congestion

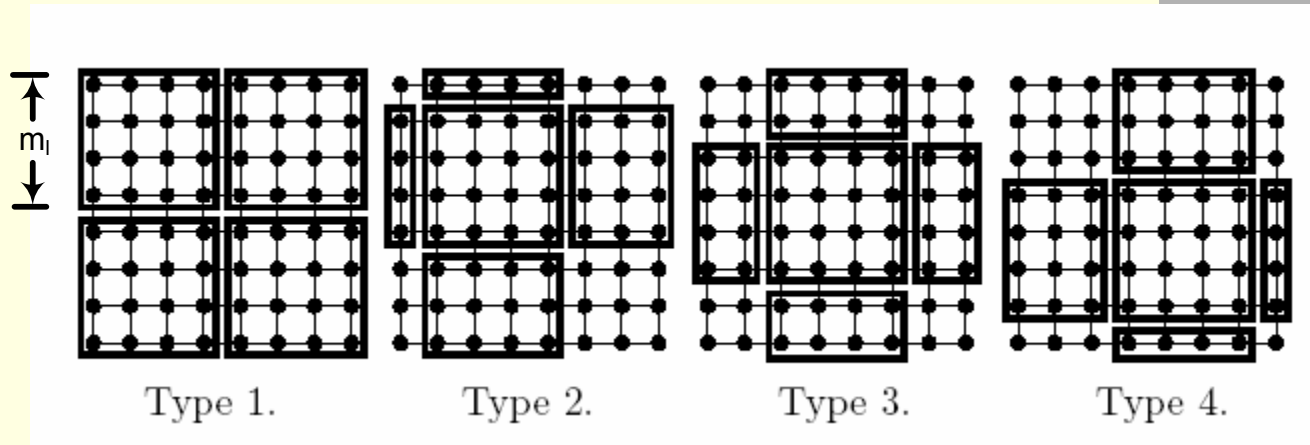
- **Lemma** $E[C(e)] \leq 16C^*(\log D^* + 3)$.

- **Theorem** $C = O(C^* \log n)$ with high probability.

Analysis on d-dimensional mesh

- The 2-dimensional decomposition can be directly generalized to a d-dimensional mesh but the stretch becomes $O(2^d)$.
- An another decomposition is used to control congestion in $O(C \cdot \log n)$ and stretch $O(d^2)$.

d-dimensional Mesh Decomposition



3-dimensional Mesh decomposition. Only 2 of the 3 dimensions are depicted

$O(d)$ types on each level

Set $\lambda = \max\{1, m/2^{\lceil \log(d+1) \rceil}\} = O(m/d)$

We shift the *type-1* submeshes by $(j-1)\lambda$ nodes in each dimension to get the *type-j* submeshes, for $j > 1$.

Analysis on d-dimensional Mesh

- **Theorem** (Stretch for d dimensions) For any two distinct nodes s and t of the mesh, $stretch(p(s, t)) = O(d^2)$.

- **Theorem** (Congestion for d Dimensions) $C = O(dC^* \log n)$ with high probability.

Random Choices

- Randomization is unavoidable for oblivious algorithms which obtain near optimal congestion.
- A is a *k-choices* algorithm if for every source-destination pairs (s,t), A chooses the resulting path from *k* possible different paths from s to t.
- A *k-choice* algorithm requires *log k* bits of randomization per packet to select any path

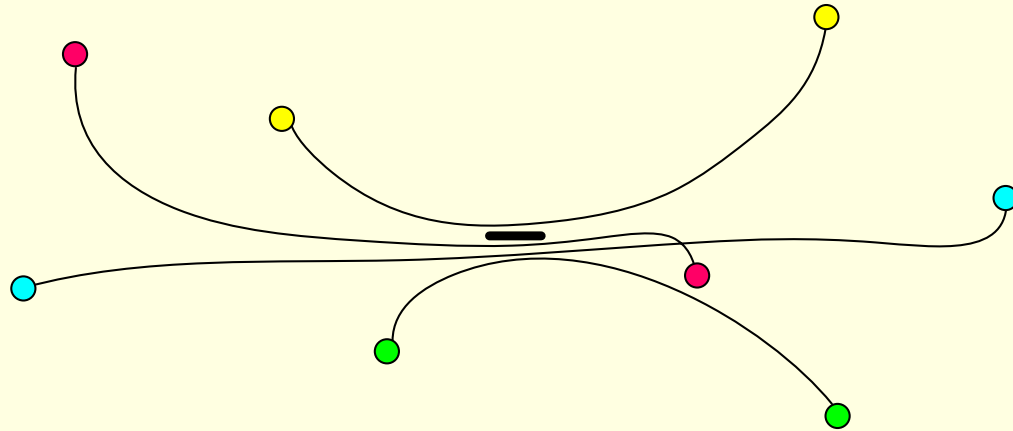
Construction of a Routing Problem

- Consider a *k-choice* algorithm A.
- Each node is the source of one packet and the destination of one packet (permutation).
- The distance from all pairs are $l = 2^s$.
- Each packet uses the path with max probability among the k possible paths provided by algorithm A.
- Average # of paths crossing an edge e is at least

$$\frac{m^d l}{dm^d} = \frac{l}{d}$$

Construction of a Routing Problem

- Π_A denote a set consisting of $\frac{l}{d}$ packets crossing e .



- **Lemma** For any *k-choice* algorithm A and corresponding routing problem Π_A with $D^* = l$, the expected congestion Y is at least $\frac{l}{dk}$.

Lower Bound on Randomization

- Lemma Let C_H denote the expected congestion of our algorithm for Π_A , then
$$C_H = O\left(\left(\frac{l}{d}\right)^{\frac{1}{d}} \cdot \log n\right).$$
- Let C_A denote the expected congestion of algo A
$$C_A \leq C_H \text{ and } \frac{l}{dk} \leq C_A \Rightarrow \text{algorithm A requires}$$
$$k = \Omega\left(\left(\frac{l}{d}\right)^{1-\frac{1}{d}} \cdot \frac{1}{\log n}\right) \text{ path choices.}$$

Lower Bound on Randomization

- Equivalently, $\Omega\left(\left(1 - \frac{1}{d}\right) \log \frac{l}{d} - \log \log n\right)$ random bits are required per packet.

$$D^* = l$$

- **Lemma** There is a routing problem with $D^* = \Omega(\log n)$ for which any algorithm A with $C_A = O(C_H)$ requires $\Omega\left(\left(1 - \frac{1}{d}\right) \log \frac{D^*}{d}\right)$ random bits per packet.

Upper Bound on Randomization

- **Lemma** For any routing algorithm, algorithm H requires $O(d \log(dD^*))$ random bits.
- **Theorem** The number of random bits used by algorithm H is within $O(d)$ of optimal.

Conclusions and Future Work

- Introduction
- Related Work
- Results on Mesh Networks
- **Conclusions and Future Work**

Conclusions

- We have shown that for Mesh, congestion and stretch can be controlled simultaneously!

	Congestion	Stretch
Maggs' algorithm	$O(dC \cdot \log n)$	$O(n)$
Our algorithm	$O(dC \cdot \log n)$	$O(d^2)$

Future Work

- Develop similar algorithm for other specific networks.
- Develop oblivious algorithms that minimize $C+D$ for general networks.



**THANK
YOU!**

