6 Routing

So far, we only concentrated on solving a fixed flow problem in a network. Now, we direct our attention to actually sending messages through the network. In practice, this is usually done by sending a message along a single path. Since many nodes may want to send out messages at the same time, the selection of the paths has to be done carefully to avoid a high congestion (and therefore a slow delivery rate). Two naive strategies would be:

- Collect information about all messages that nodes want to send out at a single node, and compute a best possible collection of paths for them.
- Run a distributed multicommodity flow algorithm in the network to converge to a near-optimal path collection, and then send out the messages.

Obviously, both strategies would create an unacceptably high delay for the messages. Ideally, we would like to have a path selection strategy that allows the nodes to decide *locally*, i.e. without consulting other nodes, along which path (resp. edge) to forward a packet. There are basically two approaches to that: *oblivious* routing and *adaptive* routing. In oblivious routing, a fixed system of optional paths is chosen *in advance* for every source-destination pair, and every packet for that pair must travel along one of these optional paths (see Figure 1). Thus, the path a packet takes only depends on its source-destination pair (and maybe a random choice to select one of the options). Formally, this can be expressed as follows:

Definition 6.1 *An oblivious routing strategy is specified by a* path system $\mathcal{P}$ *and a function* w *assigning a* weight *to every path in* $\mathcal{P}$. w *has the property that for every source-destination pair* (s,t)*, the system of flow paths* $\mathcal{P}_{s,t}$ *for* (s,t) *fulfills* $\sum_{q\in\mathcal{P}_{s,t}} w(q) = 1$.

Hence, for oblivious routing to work efficiently, we need a "good" path system.

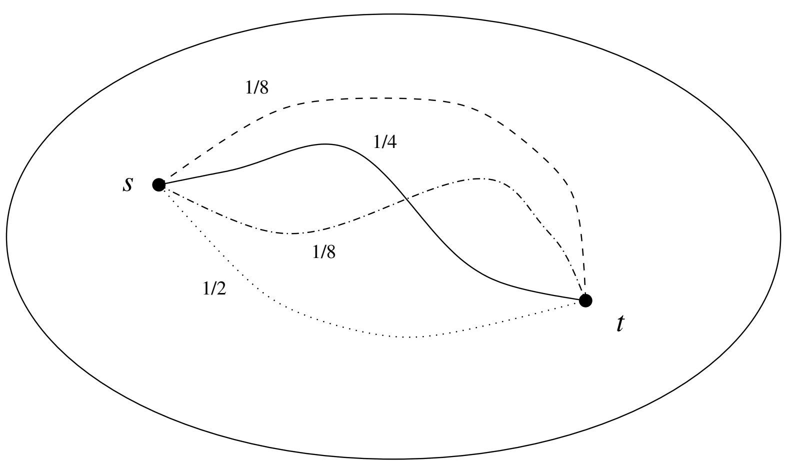

Figure 1: A system of optional paths for the pair (s,t). The numbers indicate how a flow from s to t has to be split among the paths, or equivalently, with which probability a path will have to be selected by a packet.

In adaptive routing, the path taken by a packet may also depend on other packets or events taking place in the network during its travel. However, in this section we will only concentrate on oblivious routing. Adaptive routing will be considered later in this lecture. We start with an example of how to select a good path system in a mesh, followed by a general lower bound on the congestion if every source-destination pair is just given a single path. Afterwards, we show how to get around this lower bound for the hypercube. At the end we refine the path selection problem for the mesh to be more competitive with best possible solutions than the path selection rule in the following subsection, which will demonstrate that despite the restrictive nature of oblivious routing it is a quite powerful concept.

6.1 Routing in a mesh

Consider the two-dimensional $n \times n$-mesh. Every node in this mesh has a number $(x, y) \in [n]^2$ where x represents its number in the x-dimension and y represents its number in the y-dimension. The $x - y$ *routing strategy* works as follows:

Given a packet with source-destination pair $((x_1, y_1), (x_2, y_2))$, first route the packet along the x-dimension from (x_1, y_1) to (x_2, y_1) and then along the y-dimension from (x_2, y_1) to (x_2, y_2).

This is certainly an oblivious routing strategy, since the path of a packet only depends on its source and destination. How well can this strategy now route arbitrary BMFPs? The following theorem gives an answer to this:

Theorem 6.2 *The $x-y$ routing strategy can route arbitrary BMFPs in an $n \times n$-mesh of unit-capacity edges with congestion at most $4n$ and dilation at most $2n$.*

Proof. Recall that in a BMFP every node is the source and destination of a demand of at most 4. Thus, every x-dimensional line in the mesh injects a total demand of at most $4n$, and every y-dimensional line in the mesh has to absorb a total demand of at most $4n$. When using the $x - y$ routing strategy, a total demand of at most $4n$ can therefore overlap at an edge in x-direction, and a total demand of at most $4n$ can overlap at an edge in y-direction. Hence, the maximum fraction of each demand that can be satisfied so that we obtain a feasible flow is at least $1/4n$, and therefore the congestion is at most $4n$. Since the $x - y$ routing strategy uses shortest paths and the diameter of the $n \times n$-mesh is equal to $2(n - 1)$, the dilation of the $x - y$ routing strategy can be at most $2n$. $\square$

Since the flow number of the $n \times n$-mesh is at least $2(n-1)$, the $x-y$ routing strategy therefore has a performance on BMFPs that is comparable to what can be achieved on average for BMFPs by using a *best possible* set of flow paths for each problem (see the properties of the flow number in Section 1).

6.2 The Borodin-Hopcroft lower bound

The nice property of the $x - y$ routing strategy is that it just has to specify one path for each source-destination pair. Does this suffice to obtain good oblivious routing strategies for arbitrary networks? The next theorem shows that there is a limit to this. A *permutation routing problem* is a problem in which every node is the source of exactly one source-destination pair and the destination of exactly one source-destination pair and all demands are equal to 1. Thus, the routing problem can be described by a permutation $\pi : V \to V$ on the set of nodes V.

Theorem 6.3 ([1]) *For every graph G of size n and degree d and every oblivious routing strategy using only a single path for every source-destination pair, there is a permutation π in which a node is traversed by at least $\sqrt{n/d}$ paths.*

Proof. Let $[n] = \{0, \ldots, n-1\}$ represent the set of all nodes in G and let $\mathcal{A}$ be any oblivious routing algorithm for G within our framework. Furthermore, let $\mathcal{P} = \{p_{i,j} : \; i, j \in [n]\}$ be the path system induced by $\mathcal{A}$. A node s is called a *source* for node i w.r.t. t if $p_{s,t}$ moves through i. In Figure 2, for example, s_3 is a source for i w.r.t. t.

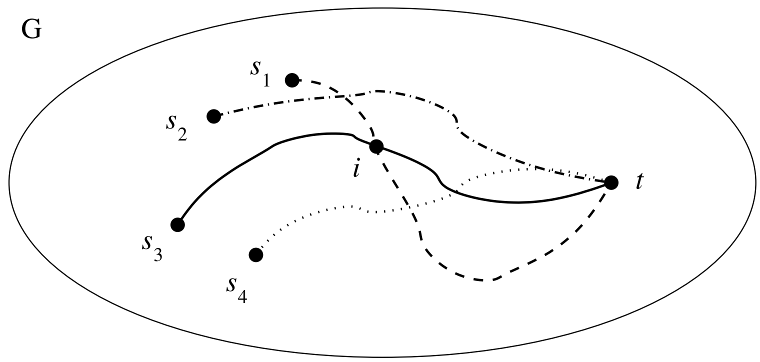

Figure 2: Illustration of the paths to k.

In the following, we will construct a permutation π with a high congestion. First we show that for any node t there are many nodes that have many sources w.r.t. t. Let $A(t, z) = \{i \in [n] : \; i$ has w.r.t. t at least z sources$\}$ be the set of all nodes that are contained in at least z different paths of $\mathcal{P}$ that lead to t. Then the following lemma holds.

Lemma 6.4 *For all $t \in [n]$: $|A(t, z)| \geq \frac{n}{d \cdot z}$.*

Proof. For any fixed $t \in [n]$, let $L = \{p_{s,t} : \; s \in [n]$ and $s \notin A(t, z)\}$, or in words, the number of paths that start outside of $A(t, z)$, and let $B \subseteq L$ be the set of all direct neighbors of nodes in $A(t, z)$ that are not in $A(t, z)$.

Obviously, $|L| = n - |A(t, z)|$. Since the maximum degree of G is d, it further holds that $|B| \leq |A(t, z)| \cdot d$. Because $B \cap A(t, z) = \emptyset$, every node in B has at most $z - 1$ paths that lead to t. Hence, $|B| \cdot (z - 1) \geq L$ and therefore

$$\begin{aligned}
& |A(t, z)| \cdot d \cdot (z-1) \geq |B| \cdot (z-1) \geq |L| \\
\Rightarrow \quad & |A(t, z)| \cdot d \cdot (z-1) \geq n - |A(t, z)| \\
\Rightarrow \quad & |A(t, z)| \cdot (d \cdot (z-1) + 1) \geq n \\
\Rightarrow \quad & |A(t, z)| \geq \frac{n}{d \cdot (z-1) + 1} \geq \frac{n}{d \cdot z} \, .
\end{aligned}$$

□

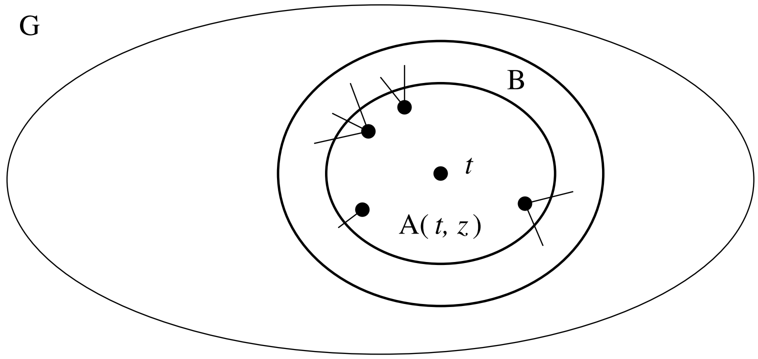

Figure 3: Illustration of $A(t,z)$ and B.

Now, let $X(z) = \{(i,t) :\ i,t \in [n] \text{ and } i \in A(t,z)\} = \bigcup_{t\in[n]}(A(t,z)\times\{t\})$. Then it holds

$$|X(z)| = \sum_{t\in[n]} |A(t,z)| \overset{\text{Lemma 6.4}}{\geq} n \cdot \frac{n}{d\cdot z} = \frac{n^2}{d \cdot z}\ .$$

For every node i let $T_i = \{t :\ (i,t) \in X(z)\}$ be the set of all destinations for which at least z paths move through i. Since

$$\sum_{i\in[n]} |T_i| = |X(z)| \geq \frac{n^2}{d\cdot z}$$

but on the other hand there are only n sets T_i, there must exist a node i with $|T_i| \geq \frac{n}{d\cdot z}$. Choose z so that z and $|T_i|$ are of the same size. This is the case for $z = \frac{n}{d\cdot z}$ or $z = \sqrt{n/d}$.

Thus, there must be a node i for which there are at least $\sqrt{n/d}$ destinations that have at least $\sqrt{n/d}$ paths through i. Simply choosing for all of these destinations one after the other any source that has not been chosen by a previous destination results in a partial permutation with an overlap of at least $\sqrt{n/d}$ paths at i. □

Thus, for constant degree networks with unit-capacity edges, the theorem implies that the congestion for routing a permutation can be as high as $\Theta(\sqrt{n})$. For unit-capacity networks with flow number $O(\log n)$ such as the butterfly this is unacceptably high, since we know from Section 1 that every BMFP and therefore also every permutation routing problem can be solved in the butterfly with congestion and dilation at most $O(\log n)$.

6.3 Valiant's Trick

We saw in Section 2.1 that oblivious routing strategies with only a single path for each source-destination pair can have an extremely high congestion. But what about multiple paths? Let $\mathcal{S}$ be the best possible solution for the multicommodity flow problem underlying the definition of F, i.e. $F = \max\{C(\mathcal{S}), D(\mathcal{S})\}$. From Theorem 4.2 we know that by applying $\mathcal{S}$ twice we can solve *every* BMFP with congestion and dilation at most $2F$. This works in a way that for every source-destination pair (s,t) we first branch off the demand from s to all other nodes in the system and afterwards reunite it at the destination t. Using $\mathcal{S}$ twice still gives an oblivious path system, but now we have many optional paths for a flow. In the case of actually sending packets, this boils down to the following strategy, which is a generalization of a well-known trick by Valiant [4]:

For every packet with source-destination pair (s, t), choose a random intermediate destination $v \in [n]$ with probability $c(v)/c(V)$ and send the packet first along a flow path in $\mathcal{S}$ from s to v and then along a flow path in $\mathcal{S}$ from v to t. (If there is more than one path from s to v resp. v to t in $\mathcal{S}$, then there will be another random experiment for picking one of these optional paths based on their flow values).

For the simple case that $\mathcal{S}$ only has a single path for every source-destination pair and $c(v)$ is the same for all nodes v, this boils down to:

For every packet with source-destination pair (s, t), choose an intermediate destination $v \in [n]$ uniformly at random and send the packet first along the path in $\mathcal{S}$ from s to v and then along the path in $\mathcal{S}$ from v to t.

To demonstrate the effect of this trick, let us consider routing in the d-dimensional hypercube. Suppose that we use for every source-destination pair $(s_{d-1}, \ldots, s_0), (t_{d-1}, \ldots, t_0) \in \{0,1\}^d$ the path that first adjusts s_0 to t_0, then s_1 to t_1, and so on, until all bits have been set to the destination's values. (I.e. these paths form our system $\mathcal{S}$.) Using Theorem 6.3, there must be a permutation with at leas $\sqrt{2^d/d}$ paths traversing a node and therefore a congestion of at least $(\sqrt{2^d/d})/d$ at an edge when using $\mathcal{S}$ directly. However, if we use Valiant's trick, we arrive at the following result.

Theorem 6.5 *Using Valiant's trick in the d-dimensional hypercube, any BMFP can be routed with congestion at most $4d$ and dilation at most $2d$.*

Proof. Since the d-dimensional hypercube has a diameter of d and $\mathcal{S}$ uses shortest paths, the dilation of Valiant's trick must certainly be at most $2d$. Thus, it remains to bound the congestion. First, consider the congestion for $\mathcal{S}$. Since every source-destination pair (or commodity) in $\mathcal{S}$ has a path of length of at most d and there are $(2^d)^2$ commodities (since there are 2^d nodes in the hypercube), the sum of the lengths of all paths in $\mathcal{S}$ is at most $d \cdot 2^{2d}$. Because the hypercube has $d \cdot 2^d/2$ edges and, due to the symmetry of the hypercube, every edge is used by the same amount of paths in $\mathcal{S}$, the number of paths overlapping at an edge is equal to

$$\frac{d\cdot^{2d}}{d \cdot 2^d/2} = 2^{d+1} . \tag{1}$$

On the other hand, the demand for every path is equal to

$$\frac{d \cdot d}{d \cdot 2^d} = \frac{d}{2^d} . \tag{2}$$

So the total demand crossing an edge is equal to $(1) \cdot (2) = 2d$, but every edge can only support a flow of 1. Hence, the maximum concurrent flow falue f for $\mathcal{S}$ is $1/(2d)$. This gives a congestion of $2d$ for $\mathcal{S}$ and therefore a congestion of $4d$ for Valiant's trick, because it doubles the overlap. $\square$

In general, it follows from Theorem 4.2:

Theorem 6.6 *For any network G with flow number F it holds: when using Valiant's trick on an optimal path collection for F, any BMFP can be routed in G with congestion and dilation at most $O(F)$.*

6.4 Oblivious routing for the mesh revisited

As we saw earlier, it is not really necessary to use Valiant's trick for the mesh to be good for all BMFPs in a sense that the congestion and dilation is always close to the flow number. However, if we are more picky here, then the $x-y$ routing strategy is still not really satisfying, since there are routing problems (other than BMFPs) where the $x-y$ routing strategy would perform very poorly. Imagine, for example, that we have a multicommodity flow problem for the $n \times n$-mesh with source-destination pairs $((i,0),(m,i))$ for all $i \in \{0,\ldots,m-1\}$, where each pair i has a demand of $d_i = m$. When using the $x-y$ routing strategy, then all paths for the pairs would go though the edge $\{(m-1,0),(m,0)\}$, causing a congestion of m^2. If, however, all pairs would have used a $y-x$ routing strategy, the congestion would have only been m (see Figure 4). In the first case it would take $\Theta(m^2)$ time steps to send a flow of m for every source-destination pair, whereas in the second case it would only take $O(m)$ steps to do this. Hence, there would be a large difference between what the $x-y$ strategy can achieve and what can be achieved in the best case. A similar counterexample can also be found for the $y-x$ strategy. Also, Valiant's trick does not help, because it would create a dilation of $\Theta(n)$, causing a time of $\Omega(n)$ to deliver all flows, whereas for the case that $m=\sqrt{n}$ this can already be achieved in $O(\sqrt{n})$ time steps. So we need a different approach.

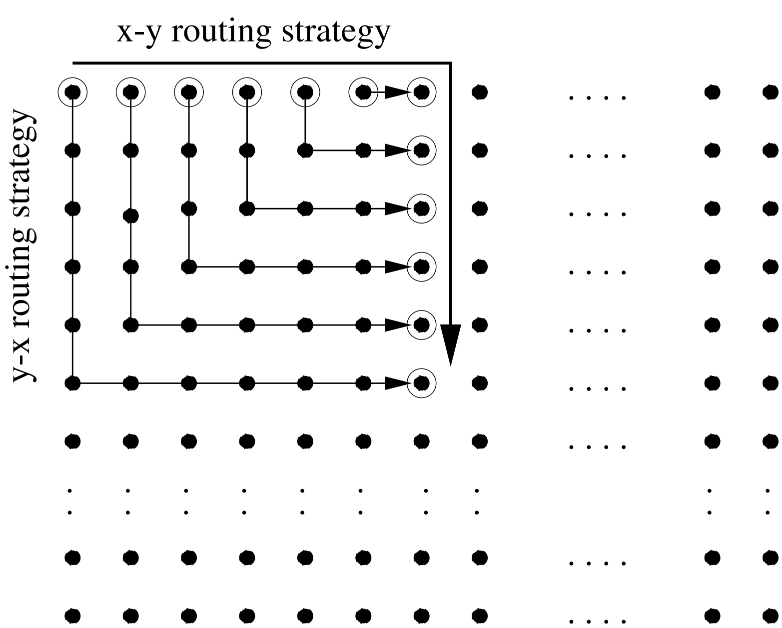

Figure 4: $x-y$ routing vs. $y-x$ routing on a mesh.

Fortunately, there is a better approach. For simplicity, we assume that we have an $n \times n$-mesh of unit-capacity edges where n is a power of 2. For every source-destination pair (s,t), a system of flow paths from s to t is recursively constructed in the following way:

Let $M_{s,t}$ be the smallest possible $2^k \times 2^k$-mesh that has s in a corner and that contains t (if this is not possible, $M_{s,t}$ represents the whole $n \times n$-mesh). The flow paths are constructed recursively as shown in Figure 5. Initially, all the flow starts at s. Then, it is evenly distributed among all nodes in M_0 using a mixed $x - y$ and $y - x$ routing strategy as sketched in Figure 5(b). That is, each node in M_0 receives a quarter of the flow, and the flow for the node at the opposite corner of s in M_0 comes in equal parts from the other two nodes in M_0. Afterwards, the flow in M_0 is evenly distributed among all nodes in M_4. Finally, the flow in M_4 is evenly distributed among all nodes in $M_{s,t}$. The same is done from t. Thus, the beginning and endpoints of the flow paths from s and t meet in $M_{s,t}$, resulting in a legal flow from s to t.

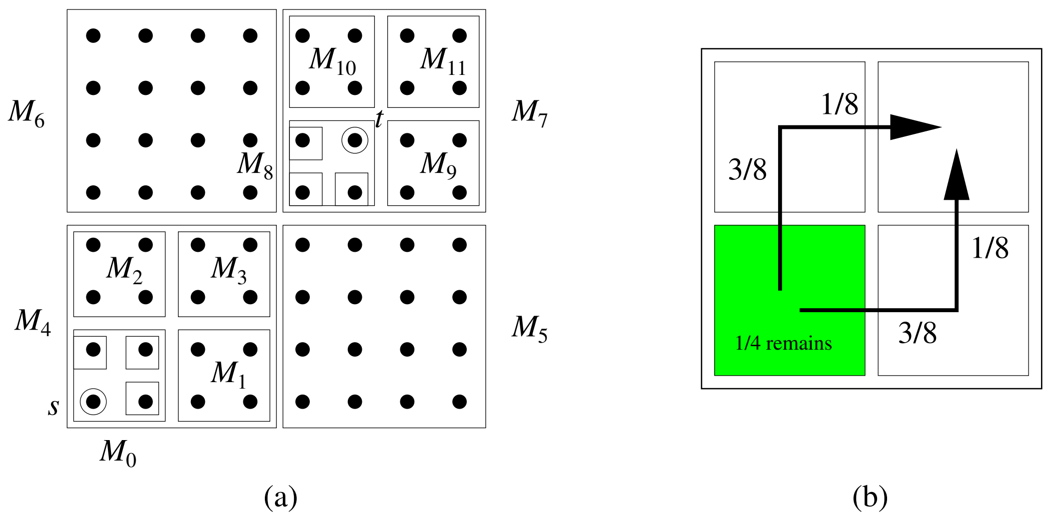

Figure 5: Recursive routing strategy from s to t. (a) illustrates the recursive decomposition into sub-meshes and (b) illustrates the distribution of flow from the shaded sub-mesh to the three other sub-meshes in its next higher mesh.

It is clear that this strategy is oblivious, but how good is it? For this we need some notation. For any multicommodity flow problem P in the $n \times n$-mesh let $C^P_{\rm OPT}$ be the best possible congestion and and $D^P_{\rm OPT}$ be the best possible dilation achievable for P (by possibly different solutions).

Theorem 6.7 *For any multicommodity flow problem P our recursive routing scheme has a congestion of $O(C^P_{\rm OPT} \cdot \log n)$ and a dilation of $O(D^P_{\rm OPT})$.*

Proof. Suppose that $M_{s,t}$ is a $2^k \times 2^k$-mesh. Then s and t must have a distance of at least 2^{k-1}. On the other side, the longest possible path our routing strategy would construct from some node s to some node t in $M_{s,t}$ is

$$2\sum_{i=0}^{k-1} 2(2^i - 1) + 2(2^k - 1) \leq 4 \cdot 2^k + 2 \cdot 2^k = 6 \cdot 2^k \; .$$

Thus, the dilation of our routing strategy is at most a constant times the maximum distance between a source-destination pair in P and therefore bounded by $O(D^P_{\rm OPT})$.

Hence, it remains to bound the congestion. Our aim will be to show that for every k, the congestion caused by all $2^k \times 2^k$-meshes used by source-destination pairs is at most $O(C^P_{\rm OPT})$. Since there are only $\log n$ different k, this results in a total congestion of $O(C_{\mathrm{OPT}}^P \cdot \log n)$. So consider some fixed k. Given a source-destination pair (s,t) with demand d, let M be a $2^k \times 2^k$-mesh that is used by (s,t) to spread its demand to all nodes in M as shown in Figure 5(b). In this case, M has a $2^{k-1} \times 2^{k-1}$-mesh M' in which the demand was initially evenly distributed among all of its nodes. That is, every node in M' had a demand of $d/2^{2(k-1)}$. When using a mixed $x-y$ and $y-x$ routing strategy for spreading it out to M, every edge is crossed by a demand of at most

$$\frac{3}{8} \cdot \frac{d}{2^{2(k-1)}} \cdot 2^{k-1} \leq \frac{d}{2^k} \ . \tag{3}$$

Now consider an edge e that is contained in m different $2^k \times 2^k$-meshes $M_1, \ldots, M_m$ that belong to source-destination pairs $(s_1,t_1), \ldots, (s_m,t_m)$ with demands $d_1, \ldots, d_m$. Then it follows from (3) that e is crossed by a total demand of at most $2^{-k} \cdot \sum_{i=1}^m d_i$. On the other hand, one can draw a $2^{k+1} \times 2^{k+1}$-mesh M around e that contains all sub-meshes M_i. Suppose that of the total demand $d = \sum_{i=1}^m d_i$ a demand of d' is routed completely inside of M from source to destination, and a demand of d'' is leaving or entering M at some point. Since the distance between s_i and t_i must be at least 2^{k-1} for every i, the average amount of the demand d' crossing an edge in M must be at least

$$\frac{2^{k-1} d'}{2 \cdot 2^{2(k+1)}} = \frac{d'}{2^{k+4}} \ .$$

Furthermore, the average amount of demand crossing an edge in $(M, \bar{M})$ must be at least

$$\frac{d''}{4 \cdot 2^{k+1}} = \frac{d''}{2^{k+3}} \ .$$

Since either d' or d'' must be at least $d/2$, *every* routing strategy must therefore have an edge that is crossed by a total demand of $\Omega(d/2^k)$, i.e. $C_{\mathrm{OPT}}^P = \Omega(d/2^k)$. On the other hand, we calculated that edge e is crossed by a demand of $O(d/2^k)$. Hence, the congestion caused by our recursive scheme is $O(C_{\mathrm{OPT}}^P)$, which completes the proof. □

This result is optimal, since it is known that for *every* oblivious routing strategy on the $n \times n$-mesh there is a routing problem P for which the strategy has a congestion of $\Omega(C_{\mathrm{OPT}}^P \cdot \log n)$ [2].

One may ask whether our upper bound for oblivious routing on a mesh can also be extended to other networks. Surprisingly, Räcke recently showed that when only looking at the congestion this is possible:

Theorem 6.8 ([3]) *For every network with non-negative capacities there is an oblivious routing strategy that achieves for every multicommodity flow problem P a congestion of $O(C_{\mathrm{OPT}}^P \cdot polylog(n))$.*

Hence, oblivious routing is a surprisingly powerful concept.

References

[1] A. Borodin and J. Hopcroft. Routing, merging, and sorting on parallel models of computation. *Journal of Computer and System Sciences*, 30:130–145, 1985.

[2] B. Maggs, F. M. auf der Heide, B. Vöcking, and M. Westermann. Exploiting locality for networks of limited bandwidth. In *Proc. of the 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 284–293, 1997.

[3] H. Räcke. Minimizing congestion in general networks. In *Proc. of the 43rd IEEE Symp. on Foundations of Computer Science (FOCS)*, 2002.

[4] L. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 2(11):350–361, 1982.